Saarland University
Department of Computer Science

Techniques for Highly Efficient and Secure Interactive Signatures

Dissertation
zur Erlangung des Grades
des Doktors der Naturwissenschaften
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

vorgelegt von

Benedikt Leo Ulrich Wagner

Saarbrücken, 2024

Tag des Kolloquiums:	02. Mai 2024
Dekan:	Univ.-Prof. Dr. Roland Speicher

Prüfungsausschuss:

Vorsitzender:	Prof. Dr. Karl Bringmann
Berichterstattende:	Dr. Julian Loss
	Prof. Dr. Markus Bläser
	Prof. Dr. Stefano Tessaro
Akademischer Mitarbeiter:	Dr. Poulami Das

Abstract

In this dissertation, we develop new techniques to construct variants of digital signatures, specifically blind signatures, multi-signatures, and threshold signatures. Our constructions are efficient and achieve strong security notions based on well-studied non-interactive assumptions.

Blind Signatures. Constructions of blind signatures are inefficient, insecure for many concurrent signing interactions, or rely on non-standard assumptions, even in the random oracle model. We design the first schemes based on conservative assumptions that are fully secure and concretely efficient.

Multi-Signatures. Existing two-round multi-signatures in the pairing-free setting and the random oracle model rely on rewinding. Thus, the concrete security level backed up by their analysis is low. We present the first constructions in this setting that avoid rewinding. For one of our constructions, security tightly relates to the underlying assumption.

Threshold Signatures. Threshold signatures in the pairing-free setting have been proven secure for static corruptions. The only exception due to Crites, Komlo, and Maller (Crypto 2023) relies on an interactive assumption and restricts the adversary to at most $t/2$ adaptive corruptions for a signing threshold t. We construct a new scheme that allows for up to t adaptive corruptions and builds on a conservative non-interactive assumption.

Zusammenfassung

In dieser Dissertation werden Techniken entwickelt, um blinde Signaturen, Multisignaturen und Thresholdsignaturen zu konstruieren. Die vorgestellten Konstruktionen sind effizient, erfüllen höchste Sicherheitsstandards und basieren auf konservativen kryptographischen Annahmen.

Blinde Signaturen. Existierende blinde Signaturverfahren sind ineffizient, erfüllen nur eine schwache Sicherheitseigenschaft oder basieren auf unkonventionellen Annahmen. In dieser Dissertation werden basierend auf konservativen Annahmen effiziente Verfahren entwickelt, die starke Sicherheitskriterien erfüllen.

Multisignaturen. Analysen existierender Multisignaturen nutzen sogenanntes Rewinding, was zu einem niedrigen konkreten Sicherheitslevel führt. In dieser Dissertation werden Konstruktionen vorgestellt, die ohne Rewinding bewiesen werden.

Thresholdsignaturen. Die Analysen von Thresholdsignaturen gelten für einen statischen Angreifer. Hierbei stellt das Verfahren von Crites, Komlo und Maller (Crypto 2023) eine Ausnahme dar, beschränkt den Angreifer jedoch auf höchstens $t/2$ adaptive Korruptionen, wobei t der Signing Threshold ist. Zudem basiert dessen Sicherheit auf einer interaktiven Annahme. In dieser Dissertation wird basierend auf einer konservativen Annahme ein Verfahren konstruiert, welches für bis zu t adaptive Korruptionen sicher ist.

Background of this Dissertation

This dissertation is based on the papers listed in the following. The individual chapters contain further information about the relation of their content and the publications. More detailed citations can be found in the Bibliography.

[CAHL+22a] **PI-Cut-Choo and Friends: Compact Blind Signatures via Parallel Instance Cut-and-Choose and More**
Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, Benedikt Wagner
CRYPTO 2022, © IACR 2022

[HLW23a] **Rai-Choo! Evolving Blind Signatures to the Next Level**
Lucjan Hanzlik, Julian Loss, Benedikt Wagner
EUROCRYPT 2023, © IACR 2023

[PW23a] **Chopsticks: Fork-Free Two-Round Multi-Signatures from Non-Interactive Assumptions**
Jiaxin Pan, Benedikt Wagner
EUROCRYPT 2023, © IACR 2023

[PW24] **Toothpicks: More Efficient Fork-Free Two-Round Multi-Signatures**
Jiaxin Pan, Benedikt Wagner
EUROCRYPT 2024, © IACR 2024

[BLT+24] **Twinkle: Threshold Signatures from DDH with Full Adaptive Security**
Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, Chenzhi Zhu
EUROCRYPT 2024, © IACR 2024

During my time as a PhD student, I also made major contributions to the following papers, which are not included in the dissertation. The list does not include the papers resulting from my Master's thesis.

[HLW23b] **Token meets Wallet: Formalizing Privacy and Revocation for FIDO2**
Lucjan Hanzlik, Julian Loss, Benedikt Wagner
IEEE S&P 2023

[PWZ23a] **Lattice-based Authenticated Key Exchange with Tight Security**
Jiaxin Pan, Benedikt Wagner, Runzhi Zeng
CRYPTO 2023

[PWZ23b] **Tighter Security for Generic Authenticated Key Exchange in the QROM**
Jiaxin Pan, Benedikt Wagner, Runzhi Zeng
ASIACRYPT 2023

[HLTW24] **Sweep-UC: Swapping Coins Privately**
Lucjan Hanzlik, Julian Loss, Sri AravindaKrishnan Thyagarajan, Benedikt Wagner
IEEE S&P 2024

[CLW24] **A Holistic Security Analysis of Monero Transactions**
Cas Cremers, Julian Loss, Benedikt Wagner
EUROCRYPT 2024

Acknowledgments

It has been a long and exciting journey from learning multiplication in elementary school to doing research in cryptography and writing this dissertation. On my way, I was lucky to meet so many amazing people, without which I would have never gotten so far. I want to take this opportunity to thank all of them, even though I will for sure not be able to mention everyone by name.

First, I want to thank my advisor Julian Loss. Thank you for trusting me to be your first Ph.D. student. Thank you for being a great mentor and thank you for every advice you gave me, be it in cryptography or life. You have always invested time and dedication to create a fruitful research environment and to guide me as a researcher. At the same time, you gave me freedom, for example, to work on the topics I find most interesting or to collaborate freely with others.

I want to thank Jiaxin Pan. You have taught me so many things about cryptography and how to write a paper. During all of our projects, you have invested so many hours in our meetings and beyond. It is always a pleasure to work with you. You also always had an open ear for a chat about life in general, and I will always remember the great and productive time we had during my visit to Norway.

Moreover, I want to express my gratitude to Mark Simkin for hosting an amazing internship at the Ethereum Foundation and my visit in Aarhus. I also wish to thank the cryptographic community, especially all of my coauthors, my colleagues at CISPA, and all the friends that I have made during conferences and research visits. I want to thank Markus Bläser and Stefano Tessaro for agreeing to review this dissertation, and Michael Reichle and Jesko Dujmovic for helpful comments on early drafts of it. Moreover, I want to thank all the teachers and professors I met on my way, who got me excited about music, maths, computer science, and cryptography.

Ich möchte die Gelegenheit ergreifen, um auch meinem privaten Umfeld von ganzem Herzen zu danken. Auch wenn es unmöglich ist, alles aufzulisten, was ich Euch verdanke, hier ein paar Beispiele: Meine Freunde sorgen für gelungene und notwendige Ablenkungen von Studium und Forschung. Meine Schwester hört sich immer meine Erklärungen zu Themen an, die mich gerade begeistern. Meine Mutter erinnert mich an die wirklich wichtigen Dinge im Leben. Meinem Vater verdanke ich die Begeisterung für Musik und Mathematik. Danke!

Meine letzte Danksagung gilt Dir, Lea. Danke für all Deine Liebe und Unterstützung. Danke, dass Du immer für mich da bist!

Contents

1

Introduction

1.1 Digital Signatures

Digital signatures [DH76] are among the most fundamental cryptographic building blocks. They allow a user, holding a secret key sk, to sign a message by computing a signature σ. Anyone can verify the signature σ using a corresponding public key pk. On the other hand, it should not be possible to generate valid signatures for new messages just given the public key pk and not sk, a property which is known as *unforgeability*. Signatures serve as an essential tool to guarantee integrity and authenticity of modern communication: we make use of digital signatures while securely browsing through the web, they can ensure authenticity of emails, and more. In recent years, the use of signatures in digital payment systems gained a lot of attention. Taking Bitcoin as an example, users may publish transactions that spend coins from one public key pk to another one. Crucially, such a transaction is only considered to be valid if it contains a valid signature with respect to pk. In this way, it is ensured that only the user holding the secret key sk for pk can spend coins associated with pk: knowing sk means owning the coins controlled by pk. Such modern applications have also motivated the definition and design of digital signature variants with enhanced functionality and privacy features. And while standard signatures are well-understood [Lam79, GMR88, NY89, Sch91, BLS01, Wat05, GPV08, Lyu12], a lot of questions remain elusive for these variants of digital signatures. In this dissertation, we develop new techniques for answering these questions.

1.2 Variants of Digital Signatures

With the advent of modern applications, such as cryptocurrencies, more expressive variants of digital signatures are needed, enhancing both functionality and privacy. In this dissertation, we study three variants, namely, blind signatures, multi-signatures, and threshold signatures.

1.2.1 Blind Signatures

In a blind signature scheme [Cha82], two parties interact to create a signature: a *Signer* holding a secret key and a *User* holding the public key and a message m to be signed. After the signing interaction, the User obtains a signature σ on the message. The scheme has to satisfy two security properties [JLO97, PS00]. On the one hand, the Signer should not be able to link the signing interaction to the pair (m, σ). In particular, the Signer does not learn anything about m during the signing interaction. This property is referred to as *blindness*. On the other hand, it is crucial that the User can not generate valid signatures without engaging with the Signer. To be precise, we insist on *one-more unforgeability*, meaning that after ℓ interactions with the Signer, the User can output signatures for at most ℓ messages. These two properties make blind signatures a highly versatile tool for creating privacy-preserving protocols. To name a few examples, blind signatures have found application in electronic cash [Cha82, OO92] (their original motivation) as well as anonymous credentials [CG08, CL01], private authentication [Goo], and electronic voting [GPZZ19]. More recently, blind signatures gained renewed interest as they facilitate private payments over public ledgers [HBG16, HLTW24].

1.2.2 Multi-Signatures

In a *multi-signature scheme* [IN83, BN06], we consider a group of signers that jointly signs a message, potentially by running an interactive signing protocol. The signature can then be verified with respect to the list of their public keys, and it certifies that *all* of the signers signed the message. The emergence of cryptocurrencies has led to renewed interest in multi-signatures [BDN18]. In this context, they can be used to create shared accounts: if multiple parties collectively possess funds, then all of them are required to sign when spending the coins. This application suggests a straightforward method for constructing multi-signatures: each signer locally signs the message, and the signature is formed by concatenating these individual signatures. The signature is considered valid if each individual signatures is valid. While this construction serves as the blueprint in terms of security, it proves inefficient as the number of signers increases. This motivates an additional criterion: *compactness*. Signatures should be much smaller than the concatenation of individual signatures. Ideally, the signature size is independent of the number of signers.

1.2.3 Threshold Signatures

As explained above, a multi-signatures requires *all* n signers to participate in the signing process, and less then n colluding signers should not be able to generate a signature that verifies with respect to the n public keys. In some scenarios, however, this requirement is too strict, for example if some signers can be temporarily unavailable. Therefore, we may relax the requirement and only ask for *enough* of the signers to participate in the signing process. More precisely, we may fix a threshold $t < n$ and allow any $t + 1$ signers to generate a signature, while no set of at most t colluding signers can. A scheme with this property is called a *threshold signature scheme* [Des88, DF90, Ped91]. As for multi-signatures, (non-trivial) threshold signatures should be compact, and the shared account application can be generalized to rely on threshold signatures. In addition to that, threshold signatures have found use in wallets [LN18], coin flipping [CKS05], and in other scenarios where trust in a single party has to be distributed [MOT+11, DOK+20].

1.2.4 Other Variants

For the interested reader, we give a non-exhaustive overview of other variants that have been studied. The referenced works may serve as a starting point for further reading. Group signatures [Cv91, Cam97, BMW03, BBS04] and (linkable) ring signatures [RST01, DKNS04, BKM06, CGS07, BKP20] allow the signer to hide its identity among a group of potential signers: anyone can verify that a member of the group signed a message, while the actual identity of the signer remains hidden. Aggregate signatures [BGLS03, LMRS04, LOS+06, HKW15] are a non-interactive version of multi-signatures allowing for potentially different messages: given triples of public keys, messages, and signatures, one can publicly, i.e., without knowing any secret key, compress the signatures into one short signature. Adaptor signatures [AEE+21, EFH+21] have found use in fair exchange protocols [TMM21, GMM+22, HLTW24]. Identity-based signatures [Sha84, DKXY03, BNN09, PW21] and forward-secure signatures [BM99, IR01, ABP13] focus on the way keys are managed and derived. Finally, one can consider combinations of the aforementioned, e.g., threshold blind signatures [CKM+23b].

1.3 Research Goal

The primary aim of this dissertation is to develop new techniques for constructing signature variants that are both *efficient* and *secure*. Subsequently, we shall expound upon the exact meaning of this.

Efficiency. To find practical applicability, cryptographic constructions must be efficient. For signatures and their variants, our primary focus encompasses four metrics of efficiency. First, we aim to minimize the *signature size* in bits as in numerous applications, signatures are subject to frequent transmission or storage. Given that the variants we study rely on interactive protocols to compute signatures, we also optimize *communication complexity*, i.e., minimize the number of bits that need to be sent. We also target a low *round complexity*. That is, each party should send a minimal number of messages during the protocol. This is because rounds introduce delay and increase the complexity of the protocol: a protocol in which each party only sends and receives one message is much simpler to implement and less error prone than a scheme with multiple communication rounds. Finally, we seek *computational efficiency* in our constructions for minimal overhead in terms of running time. Concretely, the time to generate and verify signatures should be minimal. As secondary efficiency goals, we shall keep key sizes within practical limits.

Expressive Security Notions. Before analyzing the security of cryptographic schemes, we need a precise mathematical definition of security. This entails specifying both the adversary's goal and its assumed capabilities. To illustrate this, consider standard digital signatures. A weak security notion would task the adversary to forge a signature just given the public key. However, the de-facto standard is a more robust notion called existential unforgeability under adaptive chosen-message attacks [GMR88]. It extends the adversary's capabilities by allowing it to see signatures for messages of its choice, with the goal of forging a signature for a new message. For this dissertation, we aspire to achieve the *strongest notion possible* for our constructions of signature variants. We exemplify this by examining blind signatures. Here, we want to allow the adversary to engage with the Signer in the signing protocol. We may require that the adversary completes interactions before starting new ones. While this allows to construct efficient schemes [BL13a, KLX22], enforcing such a *sequential* behavior is infeasible in practice without opening the door to denial of service attacks. Instead, we should aim for a stronger notion, in which the adversary can interact *concurrently* with the Signer, interleaving signing interactions in an arbitrary manner.

Conservative Assumptions. Security proofs in modern cryptography have a common structure: we assume the existence of an efficient adversary breaking the cryptographic scheme. Using this adversary as a subroutine, we construct an efficient algorithm that solves some computational problem Π. Under the *assumption* that Π is hard, security of the scheme follows. Of course, one could just define Π to be the problem of breaking the scheme, in which case proving security would be trivial. However, we would not gain any confidence in the security of the scheme. Instead, we want to rely on more conservative assumptions. These assumptions should possess two key characteristics: First, they should be *easy to state and non-interactive*. This means that a solver for Π gets an input and produces an output without any further interaction with the problem (e.g., no oracle is involved). Such assumptions are far more straightforward to understand and study compared to complex interactive assumptions. Second, the assumption should be *well-studied*, i.e., we have a good understanding of the

underlying mathematical structure and cryptanalysts failed to falsify the assumption despite efforts. The more conservative the underlying assumption, the greater confidence we can place in the security of our scheme. The well-known discrete logarithm and Diffie-Hellman assumptions [DH76] are examples of such conservative assumptions.

One may hope that we can avoid making an assumption in the first place. Unfortunately, it is well-known that for most cryptographic primitives, unconditional security would have unexpected implications in computational complexity theory, such as solving the longstanding **P** vs. **NP** problem [Gol01].

Concrete Security. When turning an adversary against the scheme into an algorithm solving the hard problem Π, as explained earlier, we establish a *security bound* of the form $\varepsilon \leq L \cdot \varepsilon_{\Pi}$. Here, ε is the probability that the adversary breaks the security of the scheme, ε_{Π} denotes the probability of successfully solving Π, and L is the so-called security loss. The security bound can be interpreted as a quantitative statement about the concrete security of the scheme. Namely, assuming Π is a 128-bit hard problem, we achieve a security level of $128 - \log(L)$ bits for the scheme. Conversely, if we target a 128-bit security level, we must set parameters, such as the elliptic curve groups used in the scheme, to make Π at least $128 + \log(L)$ bits hard, according to current cryptanalytic knowledge. In both cases, our objective is to minimize L. Ideally, the proof is *tight*: L is a small constant independent of any adversarial choices. A large body of research is centered around the concept of tightness: tightness has been studied for public-key encryption [BBM00, HJ12, BJLS16, GHKW16, Hof17], key exchange [BHJ+15, GJ18, LLGW20, HJK+21, DG21], as well as for digital signatures [KW03, HJ12, AFLT12, BKKP15, BJLS16, BL16, KMP16, DGJL21, PW22] and related primitives [CW13, BKP14, GHKP18, LP20]. Unfortunately, techniques used for minimizing L and achieving tightness often introduce redundancy leading to a significant efficiency overhead. In this dissertation, it is our goal to *keep the security loss minimal* without compromising efficiency.

Minimal Idealizations. In addition to hardness assumptions, numerous cryptographic proofs rely on idealizations. One prominent example is the *random oracle model*, first explicitly introduced by Bellare and Rogaway [BR93]. This model idealizes hash functions in the following way: instead of permitting parties to evaluate the function locally, every party has access to an oracle implementing a truly random function. Despite (contrived) constructions demonstrating that the random oracle can not be instantiated by a real hash function [CGH98, GK03, BBP04, BFM15], highlighting that it is not an assumption, the cryptographic community widely accepts the random oracle model. Furthermore, all practical constructions of digital signatures depend on it. Thus, we analyze our constructions in the random oracle model, but we refrain from introducing additional idealizations, such as generic or algebraic group models [Sho97, Mau05, FKL18].

1.4 Overview of Results

We introduce new construction techniques and advance the state-of-the-art in the realm of blind signatures, multi-signatures, and threshold signatures. Below, we summarize our results.

In the following, we reuse parts of the abstracts of [CAHL+22a, HLW23a, PW23a, PW24, BLT+24].

1.4.1 Results for Blind Signatures – Chapter 3

We construct the first concretely efficient blind signatures from conservative assumptions.

Background. In the random oracle model, known constructions of blind signature schemes are either prohibitively inefficient, rely on non-standard assumptions, or are only secure as long as a small number of concurrent signing interactions are allowed. Addressing this issue, Katz, Loss, and Rosenberg [KLR21] introduced a boosting transform based on a cut-and-choose technique. It converts a base scheme, secure for logarithmically many (in the security parameter) concurrent signing interactions, into a fully secure scheme, supporting any polynomial number of concurrent interactions. Unfortunately, instantiations of this transform exhibit two drawbacks. Firstly, the Signer increases a counter in every signing interaction, and the communication complexity of the resulting blind signing protocol grows linearly with this counter. Secondly, the schemes inherit a very loose security bound from the base scheme and, as a result, require impractical parameter sizes.

Contribution: PI-Cut-Choo and Friends. We eliminate these drawbacks by proposing two practical blind signature schemes, based on the RSA and CDH assumption in the pairing-setting, respectively. Their communication complexity only grows logarithmically with the number of interactions so far.

Our RSA-based scheme has signatures and communication of roughly 9 KB and 8 KB, respectively. Conceptually, we apply a communication-efficient variant of the boosting transform to the Okamoto-Guillou-Quisquater blind signature scheme [Oka93, PS96, HKL19], accompanied by careful tweaks to improve the concrete efficiency.

For our CDH-based scheme, signatures and communication are roughly 3 KB and 120 KB, respectively. Here, we use the BLS blind signature scheme [BLS01, Bol03] as a base scheme. Given that this base scheme can not tolerate any signing interaction without relying on an interactive assumption, we design a stronger variant of the boosting transform. In a nutshell, by repeating the cut-and-choose for many independent keys in parallel, we can guarantee that there is one key for which the reduction does not need to engage in any signing interaction. Exploiting aggregation features of blind BLS, we avoid most of the overhead induced by this parallel repetition. See Chapter 3 for more details.

Contribution: Rai-Choo. Our schemes mentioned above still have limitations: the Signer must maintain state, computation scales linearly with the number of signing interactions, and at least five moves of interaction are required per signature. As our second result, we introduce a scheme that eliminates all of the above drawbacks at the same time. Namely, we show a round-optimal, concretely efficient, concurrently secure, and stateless blind signature scheme in which communication and computation are independent of the number of signing interactions. It also generalizes naturally to the partially blind signature setting, where parts of the message are known to the Signer. Our scheme is based on the CDH assumption in the asymmetric pairing setting and has signature and communication sizes of 9 KB and 36 KB, respectively. To improve the (amortized) communication complexity of our scheme even further, we show how to efficiently batch the issuing of signatures for multiple messages. For more details, see Chapter 3.

1.4.2 Results for Multi-Signatures – Chapter 4

For multi-signatures, our results outperform existing constructions in the pairing-free discrete logarithm setting in terms of concrete security.

Background. Early constructions of multi-signatures in the pairing-free discrete logarithm setting require three-round signing. Recent constructions have managed to reduce the round complexity to two, i.e., to generate a signature, each signer has to send two messages. However, the security proofs of these constructions come with a significant security loss, resulting in low concrete security levels. The reason for this loss is a (nested) use of so-called rewinding, where the reduction only succeeds if the adversary succeeds in multiple dependent runs of the experiment.

Contribution: Chopsticks. We propose two efficient two-round multi-signature schemes from the (standard, non-interactive) DDH assumption in the pairing-free setting. Both schemes are proven

secure in the random oracle model without rewinding. Our first scheme supports the aggregation of keys but has a security loss linear in the number of signing queries. Our second scheme is the first tightly secure construction.

Technically, we base our constructions on a new homomorphic dual-mode commitment scheme for group elements. The commitment scheme allows to equivocate for messages of a specific structure. The definition and efficient construction of this commitment scheme is of independent interest. By combining our commitment scheme with the lossy identification technique [KW03, AFLT12, KMP16] and a guessing argument, we obtain our first scheme. To eliminate the guessing argument and achieve tight security, we introduce the pseudorandom matching technique. Here, signers hold two keys and pseudorandomly match their keys to two possible commitment keys per message. See Chapter 4 for more details.

Contribution: Toothpicks. While the Chopsticks schemes are an interesting first theoretical step, they are much less efficient than their non-tight counterparts. We close this gap by proposing a new tightly secure two-round multi-signature scheme that is as efficient as non-tight schemes. Our scheme is again based on the DDH assumption without pairings. Compared to Chopsticks, we reduce the signature size and communication complexity by more than a factor of 2, respectively. Similarly, we improve the efficiency of the non-tight scheme in Chopsticks.

To achieve this, we first, somewhat surprisingly, observe that the commitment scheme in Chopsticks does not have to be fully binding. This allows us to construct a more efficient commitment scheme. Further, we develop a new pseudorandom path technique, eliminating redundancies caused by the pseudorandom matching technique. Again, we refer to Chapter 4 for more details.

1.4.3 Results for Threshold Signatures – Chapter 5

In the realm of threshold signatures, we focus on security in the presence of adaptive corruptions.

Background. Known constructions of threshold signatures in the pairing-free discrete logarithm setting are proven secure under static corruptions. That is, the adversary is assumed to announce the set of corrupted parties ahead of time before learning any keys or signatures, and then controls these parties throughout the security experiment. What matches reality more closely is the stronger notion of security under adaptive corruptions. In this model, the adversary can corrupt parties throughout the game depending on keys, signatures, and the results of previous corruptions. Recently, Crites, Komlo, and Maller [CKM23a] have proposed Sparkle, which is the first threshold signature scheme in the pairing-free discrete logarithm setting to be proven secure under adaptive corruptions. However, without using the algebraic group model, Sparkle's proof imposes an undesirable restriction on the adversary. Namely, for a signing threshold $t < n$, the adversary is restricted to corrupt at most $t/2$ parties. In addition, Sparkle's proof relies on a strong one-more assumption.

Contribution: Twinkle. We propose Twinkle, a new threshold signature scheme in the pairing-free setting which overcomes these limitations. Twinkle is the first pairing-free scheme to have a security proof for up to t adaptive corruptions without relying on the algebraic group model. It is also the first such scheme with a security proof under adaptive corruptions from a well-studied non-interactive assumption, namely, the DDH assumption.

We achieve our result in two steps. First, we design a generic scheme based on a linear function that satisfies several abstract properties and prove its adaptive security under a suitable one-more assumption related to this function. In the context of this proof, we also identify a gap in the security proof of Sparkle and develop new techniques to overcome this issue. Second, we give a suitable instantiation of the function for which the corresponding one-more assumption follows from DDH. For more details, see Chapter 5.

1.5 Structure of this Dissertation

In Chapter 2, we recall general cryptographic background that is needed for multiple chapters of this dissertation. Then, in Chapters 3 to 5 we present our results. Each of these chapters is based on one or two publications. Namely, in Chapter 3, we present our results on blind signatures based on [CAHL+22a, HLW23a]. Chapter 4 contains our results on multi-signatures based on [PW23a, PW24]. Chapter 5 is based on [BLT+24] and contains our results on threshold signatures. Each of these three chapters has the following structure: it starts with a short publication history, explaining how it relates to the corresponding publications. Then, an introduction follows, motivating the problem, summarizing the contributions of the chapter, comparing to related work and (if any) summarizing subsequent works. After giving chapter-specific preliminaries, the main technical part with technical overviews and all constructions and proofs follows. In Chapter 6, we motivate open problems for future work and conclude.

2
General Preliminaries

2.1 Notation

In the following, we introduce general notation and conventions that are used throughout the dissertation. Many of these notations are common, see for example [KL14].

Sets. We denote the natural numbers (excluding 0) by $\mathbb{N}$, the reals by $\mathbb{R}$, and the integers by $\mathbb{Z}$. We denote the set of the first L natural numbers by $[L] = \{1, \dots, L\} \subseteq \mathbb{N}$. By $\mathbb{R}_{\geq 0}$ we denote the set of non-negative reals. For a number $n \in \mathbb{N}$, the ring of integers modulo n is denoted by $\mathbb{Z}_n = \mathbb{Z}/n\mathbb{Z}$. We naturally identify $\mathbb{Z}_n$ with the canonical set of representatives $\{0, \dots, n-1\}$. We denote by $\mathbb{Z}_n^*$ the group of units in the ring $\mathbb{Z}_n$. By $\{0,1\}^*$, we denote the set of all finite strings of bits, i.e., $\{0,1\}^* = \bigcup_{i \geq 0} \{0,1\}^i$. The cardinality of a set S is denoted by $|S|$.

Algebra. Given two numbers $a, b \in \mathbb{N}$, we write $\gcd(a,b)$ to denote the greatest common divisor of a and b. We may denote groups additively or multiplicatively, which should always be clear from the context. To denote vectors and matrices we use bold letters. By default, vectors are column vectors. For a sequences and vectors, we use subscripts to refer to specific coordinates, e.g., for $x \in \{0,1\}^*$, x_i denotes the ith bit of x. $|\mathbf{x}|$ to denote the number of coordinates of $\mathbf{x}$. Further, we use $|\cdot|$ to denote the length of a sequence or vector, e.g., for $x \in \{0,1\}^*$, $|x|$ denotes the number of bits of x. For a matrix $\mathbf{A}$, the transpose of $\mathbf{A}$ is denoted by $\mathbf{A}^t$, and the coordinate in row i and column j is denoted by $\mathbf{A}_{i,j}$.

Probability. For a finite set S, we write $x \xleftarrow{\$} S$ to state that x is sampled uniformly at random from S. For a distribution $\mathcal{D}$, we use the notation $x \leftarrow \mathcal{D}$ to indicate that x is sampled according to $\mathcal{D}$. Let $\mathbf{Exp}$ be a probabilistic experiment defining a probability space, Ev be an event, and X be a random variable that is generated during $\mathbf{Exp}$. We write $\Pr_{\mathbf{Exp}}[\mathsf{Ev}]$ or $\Pr[\mathsf{Ev} \mid \mathbf{Exp}]$ to denote the probability of Ev in the experiment $\mathbf{Exp}$. We omit $\mathbf{Exp}$ if it is clear from the context and simply write $\Pr[\mathsf{Ev}]$. We write $\{X \mid \mathbf{Exp}\}$ to denote the distribution of X in $\mathbf{Exp}$. Similarly, we may write $\mathbb{E}[X]$, $\mathbb{E}[X \mid \mathbf{Exp}]$, or $\mathbb{E}_{\mathbf{Exp}}[X]$ to denote the expectation of X. For two distributions $\mathcal{D}, \mathcal{D}'$, their statistical distance is given as $\sum_x |\Pr[\mathcal{D} = x] - \Pr[\mathcal{D}' = x]|/2$, where x ranges over all possible values that $\mathcal{D}$ and $\mathcal{D}'$ can take.

Algorithms. Unless explicitly stated otherwise, we always assume algorithms to be probabilistic. Let A be a (probabilistic) algorithm. By $\mathbf{T}(\mathsf{A})$, we denote the (upper bound) of A's running time, which is a implicitly a function of A's input length. More precisely, this means that for any input x, A is guaranteed to terminate in $\mathbf{T}(\mathsf{A})(|x|)$ steps. We say that A is PPT (probabilistic polynomial-time), if $\mathbf{T}(\mathsf{A})$ is a polynomial in $|x|$. If A is deterministic, we write $y := \mathsf{A}(x)$ to state that A is run on input x and y is assigned to the resulting output. For a probabilistic algorithm A, we instead write $y := \mathsf{A}(x; \rho)$ if the random coins that A uses are ρ. When writing $y \leftarrow \mathsf{A}(x)$, we mean that the random coins ρ are sampled uniformly at random. In other words, we treat A as a distribution. The notation $y \in \mathsf{A}(x)$ states that there are some random coins ρ such that $y = \mathsf{A}(x; \rho)$, i.e., y is a possible output of A. A cryptographic scheme is a tuple of algorithms. We may make the cryptographic scheme Sch to which an algorithm A belongs explicit by writing $\mathsf{Sch}.\mathsf{A}$ instead of A. For functions or algorithms $\mathsf{O}_1, \dots, \mathsf{O}_k$, we write $\mathsf{A}^{\mathsf{O}_1, \dots, \mathsf{O}_k}$ to indicate that A gets oracle access to $\mathsf{O}_1, \dots, \mathsf{O}_k$. That is, A can submit an input x to any O_i, and receives $\mathsf{O}_i(x)$ in return. We allow such oracles to share state.

Functions and Asymptotics. For a number $t \in \mathbb{R} \setminus \{0\}$, $\log t$ denotes the base-2 logarithm of t. As standard in theoretical computer science, we will sometimes use asymptotic notation like O, Θ, Ω, o, and ω. By λ, we denote the security parameter, and assume that every algorithm implicitly gets λ encoded in unary as 1^λ as input. We say that a function $f\colon \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligble in its input λ, if $f \in \lambda^{-\omega(1)}$. That is, it vanishes faster than the inverse of any polynomial. A function f is said to be overwhelming if $1 - f$ is negligible. We denote by poly a non-further specified polynomial function, e.g., if we write $L = \mathsf{poly}(\lambda)$, we mean that $L = L(\lambda)$ may depend on λ and there is a polynomial p such that $L \leq p(\lambda)$.

Pseudocode and Games. We specify security notions through probabilistic experiments also known as games. We denote the event that a security game $\mathbf{G}$ outputs 1 by $\mathbf{G} \Rightarrow 1$. Algorithms, oracles, and security games are specified either verbally or using pseudocode. Our pseudocode uses standard imperative programming constructs such as **if** and **else** statements and **for** loops, where we allow

loops to run over sets, implicitly assuming a natural ordering. Our pseudocode is indentation-based. Numerical values in pseudocode are implicitly initialized to 0, lists and sets to $\emptyset$, and maps initially satisfy $f[x] = \perp$ for all x. When an algorithm or an oracle terminates with an output x (which could be $\perp$), we write $\mathbf{return}\ x$. In a game $\mathbf{G}$, we may write $\mathbf{abort}$ if the entire execution of $\mathbf{G}$ stops unsuccessfully. In this case, $\mathbf{G} \Rightarrow 1$ does not hold. The notation $\mathbf{parse}\ a := b$ means that variable b is parsed according to the structure of a, e.g., we write $\mathbf{parse}\ (\mathsf{pk}, x) := b$ to denote that variable b is parsed as a tuple (pk, x). Implicitly, this means that the algorithm or game aborts if parsing fails, i.e., if b does not have this form. Multiple of these statements may be grouped together, e.g., $\mathbf{parse}\ a := b, c := d$, which means that we parse b into a and d into c.

2.2 Methodology

In this dissertation, we define security using game-based security notions, use the random oracle model, and follow the asymptotic security paradigm while precisely stating security bounds. We provide more details on this in the following paragraphs.

Game-Based Security. In this dissertation, all security notions will be game-based, as opposed to simulation-based. This is the de-facto standard of modeling security for variants of digital signatures, with a only few exceptions [Fis06, Lin22].

Random Oracle Model. We analyze our constructions in the random oracle model [BR93], which idealizes hash functions as truly random functions that are given to algorithms as an oracle. More concretely, when we say that $\mathsf{H}\colon \mathcal{X} \to \mathcal{Y}$ is a random oracle, we mean the following: both in the scheme as well as in all security games, every algorithm gets oracle access to H. In particular, the adversary gets access to H. The random oracle is then implemented using a map h in the following lazy manner: on input $x \in \mathcal{X}$, the oracle first checks if $h[x] = \perp$. If so, it samples a hash value $h[x] \xleftarrow{\$} \mathcal{Y}$ uniformly at random. In any case, it returns $h[x]$ as the hash value. Further, in security proofs, reductions simulate the random oracle for the adversary. Therefore, reductions can tamper with the way the random oracle is implemented, as long as this is indistinguishable from the honest implementation. Let us fix some conventions that we will use throughout the dissertation. First, we will generally assume that the domain is $\mathcal{X} = \{0,1\}^*$, i.e., any string can be input into the random oracle. This simplifies notation and is in line with how actual hash functions work. At the same time, our proofs often assume that inputs can be parsed, i.e., they have a certain structure. For example, we may assume a random oracle $\mathsf{H}\colon \{0,1\}^* \to \mathcal{Y}$ but then interpret inputs as pairs $(\mathsf{pk}, \mathsf{m})$ of public keys and messages. This is justified by noticing that as long as the length (in binary representation) of the components is fixed[1], parsing is non-ambiguous, and any query that can not be parsed can be treated independently and is irrelevant to the scheme. Second, to simplify notation, we generally assume multiple random oracles. This can easily be implemented from one random oracle (i.e., one hash function in practice) by proper domain separation.

Concrete and Asymptotic Security. Formally, we follow the asymptotic security paradigm. Namely, as explained earlier, we assume that there is a security parameter λ and we generally treat a scheme as secure if the respective advantage of any PPT adversary is negligible as a function of λ. At the same time, we always precisely state our security bounds, e.g., dependent on the number of random oracle queries. This makes it easy to interpret our results from the standpoint of concrete security, which is in line with the research goals we specified in Section 1.3. For a discussion on concrete vs. asymptotic security, we refer the interested reader to [KL14].

[1] For signature schemes and their variants, we can assume a fixed length for messages without loss of generality.

2.3 Cryptographic Assumptions

In this section, we introduce relevant algebraic structures and computational hardness assumptions.

Assumptions in Pairing-Free Groups. In Chapters 4 and 5, we rely on assumptions in (families of) cyclic groups. We define our assumptions with respect to a PPT algorithm GGen generating these groups. That is, GGen takes as input the security parameter λ in unary and outputs the description of a group $\mathbb{G}$ of prime order p with generator g. The description defines how elements are represented and how operations on the group are carried out. In general, we denote the group operation for groups output by GGen multiplicatively. Only when groups are interpreted as abstract vector spaces, which will be clear from the context, we prefer additive notation. Throughout, we make several natural assumptions about efficiency of operations in such groups. All of these assumptions hold true for groups used in practice, e.g., based on elliptic curves, see [KL14] for a discussion. First, we assume that group membership is always implicitly and efficiently tested. That is, if we say a party or algorithm outputs an element $h \in \mathbb{G}$, then we require any implementation to verify that the bitstring representing h indeed represents a valid group element. Second, we assume that the group operation, finding inverses, and testing equality is efficient. We also assume that randomly sampling a group element $h \xleftarrow{\$} \mathbb{G}$ is efficient. Third, we allow random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$ with range $\mathbb{G}$. Note that this means that we implicitly assume that one can hash obliviously into the $\mathbb{G}$. Especially, when computing a group element $h = \mathsf{H}(s)$ for some $s \in \{0,1\}^*$, one does not learn the discrete logarithm $x \in \mathbb{Z}_p$ such that $h = g^x$. Subsequently, we define the relevant cryptographic assumptions in such cyclic groups, namely the discrete logarithm and Diffie-Hellman assumptions [DH76]. We start with the discrete logarithm (DLOG) assumption, which states that for a random exponent $x \in \mathbb{Z}_p$ it is hard to compute x given g^x.

Definition 2.1 (DLOG Assumption)**.** *We say that the* DLOG *assumption holds relative to* GGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{DLOG}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Pr\left[\mathcal{A}(\mathbb{G},p,g,g^x) = x \mid (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda),\ x \xleftarrow{\$} \mathbb{Z}_p\right].$$

Next, we recall the Decisional Diffie-Hellman (DDH) assumption. It states that it is hard to distinguish $(g,h,u,v) \in \mathbb{G}^4$, where $h,u,v \in \mathbb{G}$ are uniform, from $(g,h,g^a,h^a) \in \mathbb{G}^4$, where $a \in \mathbb{Z}_p$ is uniform.

Definition 2.2 (DDH Assumption)**.** *We say that the* DDH *assumption holds relative to* GGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Big|&\Pr\left[\mathcal{A}(\mathbb{G},p,g,h,g^a,h^a) = 1 \,\middle|\, (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda),\ h \xleftarrow{\$} \mathbb{G},\ a \xleftarrow{\$} \mathbb{Z}_p\right]\\ &- \Pr\left[\mathcal{A}(\mathbb{G},p,g,h,u,v) = 1 \,\middle|\, (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda),\ h,u,v \xleftarrow{\$} \mathbb{G}\right]\Big|.\end{aligned}$$

In Chapter 4, we will use two variants of DDH, namely, the multi-instance variant Q-DDH and a variant we call $\mathsf{uDDH3}$. Both are (tightly) equivalent to DDH. Namely, Q-DDH is tightly implied by DDH using random self-reducibility [EHK+13]. Further, $\mathsf{uDDH3}$ is the 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption, with terminology as in [EHK+13]. By its random self-reducibility (Lemma 1 in [EHK+13]), this 2-fold $\mathcal{U}_{3,1}$-Matrix-DDH (MDDH) assumption is tightly equivalent to the $\mathcal{U}_{3,1}$-MDDH assumption. By Lemma 1 in [LP20], $\mathcal{U}_{3,1}$-MDDH is tightly equivalent to $\mathcal{U}_1$-MDDH, which is the DDH assumption. Hence, the DDH and $\mathsf{uDDH3}$ assumptions are tightly equivalent.

Definition 2.3 (Q-DDH Assumption)**.** *We say that the* Q-DDH *assumption holds relative to* GGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{Q\text{-}\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Bigg|&\Pr\left[\mathcal{A}\left(\mathbb{G},p,g,h,(g^{a_i},h^{a_i})_{i=1}^Q\right) = 1 \,\middle|\, \begin{array}{l}(\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda),\\ h \xleftarrow{\$} \mathbb{G},\ \forall i \in [Q]:\ a_i \xleftarrow{\$} \mathbb{Z}_p\end{array}\right]\\ &- \Pr\left[\mathcal{A}\left(\mathbb{G},p,g,h,(u_i,v_i)_{i=1}^Q\right) = 1 \,\middle|\, \begin{array}{l}(\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda),\\ h \xleftarrow{\$} \mathbb{G},\ \forall i \in [Q]:\ u_i,v_i \xleftarrow{\$} \mathbb{G}\end{array}\right]\Bigg|.\end{aligned}$$

Definition 2.4 (uDDH3 Assumption)**.** *We say that the* uDDH3 *assumption holds relative to* GGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \left| \Pr\left[\mathcal{A}(\mathbb{G},p,g,(h_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l} (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda), \\ a,b \xleftarrow{\$} \mathbb{Z}_p,\ h_{1,1},h_{2,1},h_{3,1} \xleftarrow{\$} \mathbb{G} \\ h_{1,2} := h_{1,1}^a, h_{1,3} := h_{1,1}^b \\ h_{2,2} := h_{2,1}^a, h_{2,3} := h_{2,1}^b \\ h_{3,2} := h_{3,1}^a, h_{3,3} := h_{3,1}^b \end{array} \right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(\mathbb{G},p,g,(h_{i,j})_{i,j\in[3]}) = 1 \;\middle|\; \begin{array}{l} (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda), \\ \forall (i,j) \in [3]\times[3] : h_{i,j} \xleftarrow{\$} \mathbb{G} \end{array} \right] \right|.$$

We also introduce AOMCDH, a new assumption (originally introduced in [BLT+23, BLT+24]) which can be understood as an algebraic one-more variant of the Computational Diffie-Hellman (CDH) assumption [DH76]. Roughly, given t-time access to an algebraic DLOG oracle, one has to solve CDH $t+1$ times. This assumption is the only interactive assumption that we use in this dissertation. We will use it to instantiate our threshold signature scheme in Chapter 5. However, this instantiation will only be a first step, and we also present a more sophisticated instantiation based on DDH, which is in line with our goal of avoiding interactive assumptions, see Section 1.3. To gain confidence in this new assumption, we also sketch how it reduces to . Again, we highlight that this is not contradicting our goals outlined in Section 1.3, as we will only use AOMCDH as a first step, and later replace it by DDH.

Definition 2.5 (AOMCDH Assumption)**.** *We say that the t-*AOMCDH *assumption holds relative to* GGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{t\text{-}\mathsf{AOMCDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda) := \Pr\left[\forall i \in \{0\}\cup[t] :\ X_i' = h^{x_i} \;\middle|\; \begin{array}{l} (\mathbb{G},g,p) \leftarrow \mathsf{GGen}(1^\lambda), \\ h \xleftarrow{\$} \mathbb{G},\ x_0,\ldots,x_t \xleftarrow{\$} \mathbb{Z}_p, \\ \forall i \in \{0\}\cup[t] : X_i := g^{x_i}, \\ (X_i')_{i=0}^t \leftarrow \mathcal{A}^{\textsc{Inv}}(\mathbb{G},g,p,h,(X_i)_{i=0}^t) \end{array} \right],$$

where $\mathcal{A}$ can call Inv *up to t times, and* Inv$(\alpha_0,\ldots,\alpha_t)$ *returns $\sum_{i=0}^t \alpha_i x_i$.*

Note that AOMCDH is different from the one-more variant introduced in [BFP21] in the sense that the adversary has an algebraic DLOG oracle instead of a CDH oracle. Next, we sketch that AOMCDH is implied by the algebraic one-more DLOG (AOMDL) assumption [NRS21] in the algebraic group model (AGM) [FKL18]. Note that Bauer et al. [BFP21] gave a bound for one-more DLOG in the generic group model (GGM) [Sho97]. As an immediate consequence, the advantage of any adversary against AOMCDH is bounded by the same, namely, by $\Theta\left((t^2/(p-t^2)) + (1/p)\right)$.

Lemma 2.1 (Informal)**.** *The* AOMCDH *assumption is implied by the algebraic one-more* DLOG *assumption in the algebraic group model.*

Proof Sketch. We only sketch the proof. Recall the AOMCDH game for an algebraic adversary $\mathcal{A}$:

1. The game generates group parameters $(\mathbb{G},g,p)$ and samples $h \xleftarrow{\$} \mathbb{G}$ and $\mathbf{x} = (x_i)_{i=0}^t \xleftarrow{\$} \mathbb{Z}_p^{t+1}$. We let $\gamma \in \mathbb{Z}_p$ be such that $h = g^\gamma$.
2. The game defines $X_i := g^{x_i}$ for all i and runs $\mathcal{A}$ on input $\mathbb{G},g,p,h,(X_i)_{i=0}^t$ with t-time access to an oracle Inv which on input $\alpha_0,\ldots,\alpha_t$ outputs $\sum_{i=0}^t \alpha_i x_i$.
3. When $\mathcal{A}$ terminates, it outputs $t+1$ group elements $(X_i')_{i=0}^t$ and wins the game if $X_i' = h^{x_i}$ for all i. As $\mathcal{A}$ is algebraic (i.e., we are in the algebraic group model), we can assume that it also outputs elements $\mathbf{a} = (a_i)_i \in \mathbb{Z}_p^{t+1}$, $\mathbf{b} = (b_i)_i \in \mathbb{Z}_p^{t+1}$, and $\mathbf{C} = (C_{i,j})_{i,j} \in \mathbb{Z}_p^{(t+1)\times(t+1)}$ such that for all i we have
$$X_i' = g^{a_i} \cdot h^{b_i} \cdot \prod_{j=0}^t X_j^{C_{i,j}}.$$

Reading these equations in the exponent of g and assuming that $\mathcal{A}$ wins, this means that

$$\gamma\mathbf{x} = \mathbf{a} + \gamma\mathbf{b} + \mathbf{C}\mathbf{x}.$$

We want to argue that $\mathcal{A}$ can not win the game. For that, we will distinguish two cases and bound the probability of these using the AOMDL and the plain DLOG assumption, respectively.

First Case: $\mathbf{b} = \mathbf{x}$. In this case, it is clear that a reduction can solve AOMDL. Before we sketch the reduction, we informally recall the AOMDL game, as present in recent works, e.g., [NRS21]. The game is exactly[2] as the AOMCDH game, with two modifications: (1) there is no element h, and (2) the winning condition asks the adversary to outputs the x_i instead of group elements X_i'. Now, our reduction is as follows. It gets as input $(\mathbb{G}, g, p)$ and $(X_i)_{i=0}^t$, it samples $h \xleftarrow{\$} \mathbb{G}$, and runs $\mathcal{A}$ on input $\mathbb{G}, g, p, h$, $(X_i)_{i=0}^t$. To answer $\mathcal{A}'s$ queries to Inv, it simply forwards them to its own oracle provided by the AOMDL game. When $\mathcal{A}$ outputs $(X_i')_{i=0}^t$ and $\mathbf{a}, \mathbf{b}, \mathbf{C}$, the reduction simply outputs $\mathbf{b}$ as a solution to the AOMDL game.

Second Case: $\mathbf{b} \neq \mathbf{x}$. In this case, we know that there is a position $i^* \in \{0\} \cup \{t\}$ such that $x_{i^*} \neq b_{i^*}$. Rearranging the i^*th equation we get

$$\gamma = \frac{a_{i^*} + \sum_{j=0}^{t} c_{i^*,j} x_j}{x_{i^*} - b_{i^*}},$$

where we used that $x_{i^*} \neq b_{i^*}$. With this, we can construct a reduction solving the plain DLOG problem: it gets as input $(\mathbb{G}, g, p)$ and $h = g^\gamma$, samples the x_i and simulates the AOMCDH game for $\mathcal{A}$, simulating Inv honestly using the x_i. When $\mathcal{A}$ outputs $(X_i')_{i=0}^t$ and $\mathbf{a}, \mathbf{b}, \mathbf{C}$, the reduction outputs γ computed as above. □

Assumptions in Pairing-Friendly Groups. In Chapter 3, we make use of assumptions in cyclic groups that are equipped with a pairing. A pairing is a map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1, \mathbb{G}_2$ are cyclic groups of prime order p with generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, respectively, and $\mathbb{G}_T$ is also a cyclic group of order p. The map is required to be bilinear, i.e., for every $a, b \in \mathbb{Z}_p$, we have $e\left(g_1^a, g_2^b\right) = e\left(g_1, g_2\right)^{ab}$, and non-degenerate, i.e., $g_T := e\left(g_1, g_2\right)$ is a generator of $\mathbb{G}_T$. Throughout, we let PGGen be an algorithm that takes as input the security parameter λ in unary and outputs the description of such pairing-friendly groups, where we distinguish three settings following [GPS08]:

- *Type-1.* We have $\mathbb{G}_1 = \mathbb{G}_2$. In other words, PGGen outputs $(\mathbb{G}, g, p, e)$ where g generates $\mathbb{G}$, $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and $\mathbb{G}_T$ is implicitly given by its generator $g_T = e\left(g, g\right)$.
- *Type-2.* We have $\mathbb{G}_1 \neq \mathbb{G}_2$ and PGGen outputs $(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e)$, $\mathbb{G}_T$ is implicitly given by its generator $g_T = e\left(g_1, g_2\right)$, and there is an efficiently computable homomorphism $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$.
- *Type-3.* We have $\mathbb{G}_1 \neq \mathbb{G}_2$ and PGGen outputs $(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e)$, $\mathbb{G}_T$ is implicitly given by its generator $g_T = e\left(g_1, g_2\right)$, and there is no efficiently computable homomorphism $\phi\colon \mathbb{G}_2 \to \mathbb{G}_1$.

Further, the type-1 setting is often called the *symmetric setting*, whereas the type-2 and type-3 setting are referred to as the *asymmetric setting*. We assume that one can efficiently identify, invert, and sample elements, test elements for equality, hash into groups, and compute the group operation similar to our assumptions in the pairing-free setting. In this dissertation, we primarily use the type-1 and type-3 setting, where we use the type-1 setting for ease of presentation and the type-3 setting as it is practically the most relevant one. It will always be clear from the context which setting we consider. For a detailed discussion on the different caveats of these settings both in theory and practice, we refer to [GPS08]. In the following, we define the Computational Diffie-Hellman (CDH) assumption [DH76] relative to PGGen in type-1 and type-3 settings. In the type-1 setting, CDH is exactly as in the pairing-free setting, namely, given g^x and g^y, CDH states that it is hard to compute g^{xy}.

[2] In the game as defined in [NRS21], the adversary gets its challenges X_i via an oracle, in which case our reduction also works.

Definition 2.6 (CDH Assumption, Type-1 Setting)**.** *We say that the* CDH *assumption holds relative to* PGGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{PGGen}}(\lambda) := \Pr\left[z = xy \;\middle|\; (\mathbb{G}, g, p, e) \leftarrow \mathsf{PGGen}(1^\lambda), x, y \xleftarrow{\$} \mathbb{Z}_p, g^z \leftarrow \mathcal{A}(\mathbb{G}, g, p, e, g^x, g^y)\right].$$

In the type-3 setting, CDH states that given g_1^x, g_1^y, and additionally g_2^y, it is hard to compute g_1^{xy}. This assumption is sometimes called co-CDH [CHKM10].

Definition 2.7 (CDH Assumption, Type-3 Setting)**.** *We say that the* CDH *assumption holds relative to* PGGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{PGGen}}(\lambda) := \Pr\left[\begin{array}{c|l} z = xy & (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda), \\ & x, y \xleftarrow{\$} \mathbb{Z}_p,\ X_1 := g_1^x,\ X_2 := g_2^x,\ Y := g_1^y \\ & g_1^z \leftarrow \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e, X_1, Y, X_2) \end{array}\right]$$

Assumptions in the RSA Setting. In Chapter 3, we make use of the RSA assumption introduced by Rivest, Shamir, and Adleman [RSA78]. Here, we again consider an algorithm that gets as input the security parameter λ in unary and outputs parameters for the assumption. Namely, we let RSAGen be an algorithm that outputs a quadruple (n, p, q, e), where p and q are distinct primes and $n = pq$, and $e \in \mathbb{N}$ satisfies $\gcd(e, \varphi(n)) = 1$. When describing algorithms and games in this setting, we assume that operations are carried out modulo n. Given n, operations, inverses, equality, and sampling over the groups $\mathbb{Z}_n$ and $\mathbb{Z}_n^*$ are efficient. The RSA assumption informally states that it is hard to compute eth roots in $\mathbb{Z}_n^*$.

Definition 2.8 (RSA Assumption)**.** *We say that the* RSA *assumption holds relative to* RSAGen*, if for all PPT algorithms $\mathcal{A}$, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{RSA}}_{\mathcal{A},\mathsf{RSAGen}}(\lambda) := \Pr\left[x^e = y \;\middle|\; (n, p, q, e) \leftarrow \mathsf{RSAGen}(1^\lambda),\ \bar{x} \xleftarrow{\$} \mathbb{Z}_n^*,\ y := \bar{x}^e,\ x \leftarrow \mathcal{A}(n, e, y)\right].$$

3

Blind Signatures

Publication History of This Chapter

This chapter is based on the two publications [CAHL+22a] and [HLW23a] and their respective full versions [CAHL+22b] and [HLW22]. I contributed to [CAHL+22a] as one of two main authors, which is due to a merge of the two papers [WHL22] and [CAL22], where I was the main author of [WHL22]. Consequently, only the parts of [CAHL+22a] for which I was responsible are included, namely, the generic transform given in [CAHL+22a] that resulted from [CAL22] is not included. I am also the main author of [HLW23a]. To obtain a consistent structure, the content of the original publications has been reordered and partially merged. Additionally, minor notational changes have been made.

3.1 Introduction

Blind signatures, introduced by David Chaum in 1982 [Cha82], are an interactive type of signature scheme with special privacy features. Informally, in a blind signature scheme, a Signer, holding a secret key sk, and a User, holding a corresponding public key pk and a message m, engage in a two-party protocol. At the end of the interaction, the User obtains a signature σ on m that can be verified using pk. Informally, a blind signature scheme must satisfy two security requirements [JLO97, PS00]. *Blindness* ensures that the Signer learns no information about m, specifically, the Signer can not link (m, σ) to the signing interaction. On the other hand, *one-more unforgeability* guarantees that the User can not obtain valid signatures without interacting with the Signer. These properties make blind signatures a useful building block for privacy-sensitive applications, e.g., e-cash [Cha82, OO92], anonymous credentials [CG08, CL01], e-voting [GPZZ19], and blockchain-based systems [HBG16].

Unfortunately, even in the random oracle model, existing constructions of blind signatures either rely on non-standard assumptions [Bol03, BNPS03, FHS15], or have parameters (e.g., communication and signature sizes) that grow linearly in the number of concurrent signing interactions [PS00, HKL19, KLR21].

Our Goal. The main goal of this chapter is to construct blind signature schemes in the random oracle model that do not suffer from any of the drawbacks mentioned above. More concretely, we will construct the first practically efficient and concurrently secure blind signature schemes from well-established non-interactive assumptions.

3.1.1 Background: Boosting via Cut-and-Choose

A long line of work [PS00, AO00, Abe01, HKL19, HKLN20] has explored constructions of blind signatures from witness indistinguishable linear identification schemes such as the Okamoto-Schnorr and Okamoto-Guillou-Quisquater schemes [Oka93]. The resulting schemes are proven secure under well-understood assumptions, such as RSA or DLOG, as long as only a polylogarithmic number of concurrent signing interactions is allowed. Some of these schemes even admit an efficient attack [Sch01, Wag02, BLL+21] if the number of concurrent singing interactions ever exceeds a polylogarithmic bound.

Inspired by an early work by Pointcheval [Poi98], Katz, Loss, and Rosenberg [KLR21] recently introduced a boosting transform that uses cut-and-choose to turn such blind signature schemes into fully secure ones, admitting a polynomial number of concurrent signing interactions. As a result, one obtains schemes that rely on well-studied assumptions and have short signatures. Unfortunately, the resulting communication and computational complexity renders them impractical. This is because in the Nth interaction between Signer and User, the communication and computation depend linearly on N. A second drawback of the transform is the loose security bound resulting in impractical parameter sizes. Indeed, for an adversary with advantage ϵ against the resulting blind signature scheme, the number of times that the reduction invokes the underlying signing oracle behaves as $\ln(1/\epsilon)$, which is large for small ϵ. Therefore, the underlying scheme already has to support a large number of signatures (if ϵ is small), requiring impractical parameter sizes. To highlight this, we computed the parameter sizes for the instantiations of the boosting transform based on the discrete logarithm problem[1]. Our calculations show that in order to support 2^{30} signatures, the scheme requires a 12035 bit group. It is apparent that this group size is impractical, and no standardized elliptic curve groups of this size exist. We remark that Katz et al. [KLR21] also provide a parameter estimate, but this holds only for a very specific choice of signing queries, random oracle queries, and advantage.

[1]We used a Python script to compute these numbers, which can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.

3.1.2 Contribution: PI-Cut-Choo and Friends

As our first contribution in this chapter, we propose two concrete blind signature schemes based on RSA and CDH, respectively, for which the communication only grows logarithmically in N and with concretely efficient parameters. In a nutshell, we first develop new ideas to eliminate the linearly growing communication of the boosting transform [KLR21] while still preserving all of its advantages. Then, we develop several techniques that are specific to the assumption that is used and improve concrete security bounds and efficiency. We summarize the efficiency of our schemes in Table 3.1.

Scheme	Signatures	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$	a	b	Max
PI-Cut-Choo	2^{20}	10.81	3.16	3.05	26.50	87.50
PI-Cut-Choo	2^{30}	11.49	3.16	3.05	26.73	118.20
PI-Cut-Choo's Friend	2^{20}	18.37	7.91	0.02	7.11	7.51
PI-Cut-Choo's Friend	2^{30}	18.74	8.66	0.02	7.48	8.08

Table 3.1: Concrete efficiency of our schemes supporting a given number of signatures and 128 bit security. Here, communication complexity is given as $a \cdot \log(N) + b$, where N is the number of issued signatures so far. Column Max shows the communication complexity for the maximum N. All sizes are in kilobytes.

PI-Cut-Choo. Our scheme from CDH (in the type-1 pairing setting) is statistically blind against malicious signers and builds on Boldyreva's blind version of the BLS signature scheme [Bol03]. Using only CDH (in contrast to a one-more variant of it), Boldyreva's scheme does not support any invocation of the signing oracle, i.e., it only provides key-only security. To account for that, we introduce a stronger variant of the boosting transform. Namely, we observe that by running the transform for several independent keys in parallel, a reduction to key-only security is possible. This also improves the concrete security bound and makes it possible to use a standard sized group. To reduce the cost of this parallel repetition, we use the aggregatability of the BLS scheme. Overall, our scheme from CDH supports 2^{30} signatures at a size of 3 KB and 120 KB communication per signature.

PI-Cut-Choo's Friend. Our scheme from RSA follows our (single instance) communication-efficient boosting transform with the Okamoto-Guillou-Quisquater (OGQ) blind signature [Oka93, HKL19] as a base scheme. It supports 2^{30} signatures at a more balanced size of 9 KB per signature and 8 KB communication per signature. To circumvent a loose security bound and maximize concrete efficiency, we use several tricks specific to the RSA setting, e.g., the existence of a trapdoor.

Semi-Honest Signer Blindness. In terms of blindness, previous works building on OGQ [HKL19, KLR21] only show honest signer blindness, i.e., the Signer's public key is generated honestly by the experiment. We first observe that our RSA-based scheme satisfies an intermediate notion we call semi-honest signer blindness, where the Signer provides the random coins to generate the public key to the experiment. We then show that by having the Signer prove knowledge of the random coins, we can transform any semi-honest signer blind scheme into a scheme that satisfies blindness against malicious signers. In the case of our RSA-based scheme, we can get malicious signer blindness either by relying on generic proof systems, or on more efficient ones based on quadratic residuosity [GRSB19] or discrete logarithms [CM99]. We emphasize, however, that using generic proofs may be sufficiently efficient in our context, as the proofs only have to be generated and verified once upon registering the Signer's public key. Especially, they do not affect the complexity of the signing protocol or the size of our signatures.

Limitations. We briefly highlight the remaining drawbacks of our first two schemes. The idea of the boosting transform fundamentally relies on a 1-of-N cut-and-choose where N, the number of signing interactions, grows over time. This requires to execute N copies of the base scheme and has the following implications:

- The Signer is stateful, as it has to keep track of the current value of N.
- The computation grows linearly in N for both the Signer and the User. To issue $N \approx 2^{30}$ signatures, this would require prohibitive computational effort (roughly $\sum_{i=1}^{2^{30}} i \approx 2^{59}$ operations).
- Issuing a signature requires five moves of interaction between Signer and User which is a far cry from the theoretical one-round limit achieved by some schemes [Bol03].

Thus, even though our first two schemes significantly improve over prior constructions, we aim to eliminate those drawbacks.

3.1.3 Contribution: PI-Cut-Choo evolves to Rai-Choo

In our second contribution in this chapter, we eliminate *all* of the aforementioned drawbacks of our first two schemes.

Rai-Choo. We construct a practical blind signature scheme using a new variant of the cut-and-choose technique, that is polynomially secure and does not require the Signer to keep a state. This eliminates the dependency on a counter N as in [KLR21, CAHL+22a] entirely, thereby also significantly reducing the computational complexity, see Table 3.2. Additionally, in contrast to schemes in [KLR21, CAHL+22a], our scheme is round-optimal. Our scheme is statistically blind against malicious signers. We show one-more unforgeability based on the (co)-CDH assumption in asymmetric type-3 pairing groups. One-more unforgeability holds for any (a priori unbounded) polynomial number of signing interactions. We obtain several parameter settings for our scheme, leading to a trade-off between signature and communication size, see Table 3.3. For example, we can instantiate parameters to obtain 9 KB signature size and 36 KB communication complexity. To demonstrate that our scheme is computationally efficient, we have implemented a prototype over the BLS12–381 curve. Our experiments show that signing takes less than 0.2 seconds, see Table 3.3.

	[KLR21]	PI-Cut-Choo's Friend	PI-Cut-Choo	Rai-Choo
Moves	7	7	5	2
Communication	$\Theta(\lambda N)$	$\Theta(\lambda \log N)$	$\Theta(\lambda \log N + \lambda^2)$	$\Theta(\lambda^2)$
Computation	$\Theta(N)$	$\Theta(N \log N)$	$\Theta(\lambda N)$	$\Theta(\lambda)$

Table 3.2: Comparison of number of moves, communication and computation for [KLR21] and our work in the Nth signing interaction. The security parameter is denoted by λ. Communication is given in bits, and computation is given by treating pairings, group and field operations, and hash evaluations as one unit.

			Communication with Batch Size L				Running Time	
	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$	$L=1$	$L=4$	$L=16$	$L=256$	Sign	Verify
(I)	0.14	13.98	33.20	16.98	12.92	11.65	163	54
(II)	0.14	9.41	36.21	20.11	16.08	14.82	169	36
(III)	0.14	5.71	72.79	43.97	36.77	34.52	333	22

Table 3.3: Efficiency of different parameter settings of our scheme. Sizes and times are given in kilobytes and milliseconds, respectively. Communication is amortized per message. Details can be found in Section 3.7.3.

Partial Blindness and Batching. We show that our scheme naturally generalizes to the setting of partially blind signatures. Additionally, we show how we can batch multiple signing interactions to

improve communication complexity (see also Table 3.3), and provide the first formal model and analysis for that. Batching has been used in many other contexts as well, e.g., in oblivious transfer [IKNP03, BBDP22]. We believe that batching blind signatures has a lot of natural use-cases. As an example, consider an e-cash scenario. Here, parties withdraw coins from a bank by getting blind signature for a random message. Later, the coin can be deposited by presenting the message-signature pair. Blindness ensures that the process of withdrawal is not linkable to the process of depositing. This approach is also used to do enhance the anonymity in electronic payment systems [HBG16]. We remark that it is crucial that all issued coins are of equal amount to guarantee a large anonymity set. Therefore, any user that wants to retrieve more than one coin has to interact with the bank multiple times to get multiple coins (i.e., signatures). Using batch blind signatures, these interactions can all happen in parallel, leading to improved communication and computational efficiency, as well as reduced overhead to initiate interactions.

3.1.4 More on Related Work

We discuss related work in more detail.

Impossibility. There are several impossibility results about the construction of blind signatures in the standard model [FS10, Pas11, BL13b]. Fischlin and Schröder showed that statistically blind three-move schemes can not be constructed from non-interactive assumptions under certain conditions [FS10]. Pass showed that unique round-optimal blind signatures can not be based on a class of interactive assumptions [Pas11]. Baldimtsi and Lysyanskaya showed that schemes with a unique secret key and a specific structure can not be proven secure, even under interactive assumptions [BL13b].

Sequential vs. Concurrent Security. In terms of unforgeability, one distinguishes concurrent and sequential security. For sequential security, the adversary has to finish one interaction with the Signer before initiating the following interaction. In contrast, concurrent security allows the adversary to interact with the Signer in an arbitrarily interleaved way. In practice, restricting communication with the Signer to sequential access opens a door for denial of service attacks. Therefore, concurrent security is the widely accepted notion.

Generic Constructions. One can build blind signatures generically from standard signatures and secure two-party computation (2PC), as shown by Juels, Luby and Ostrovsky [JLO97]. Fischlin [Fis06] gave a (round-optimal) generic construction that is secure even in the universal composability framework [Can00]. However, it turns out that instantiating these generic constructions efficiently is highly non-trivial. For example, instantiating Fischlin's construction requires to prove statements in zero-knowledge about a combination of commitment and signature scheme. If we instantiate the signature scheme efficiently in the random oracle model, we end up treating the random oracle as a circuit. This leads to unclear implications in terms of security. The recent work by del Pino and Katsumata [dK22] makes significant progress in this direction. By carefully choosing building blocks and slightly tweaking the construction, they give an instantiation of Fischlin's paradigm in the lattice setting. However, the communication complexity of their protocol is still far from being practical. In Section 3.1.5, we mention more recent progress in this direction.

Efficiency from Strong Assumptions. In addition to the generic constructions mentioned above, there are direct constructions of blind signatures. While some constructions make use of complexity leveraging [GRS+11, GG14], others are proven secure under non-standard q-type or interactive assumptions [Oka06, GRS+11, FHS15, Gha17]. Notably, there are efficient and round-optimal schemes based on the full-domain-hash paradigm [Bol03, BNPS03, AKSY21]. For example, Boldyreva [Bol03] introduces a blinded version of the BLS signature scheme [BLS01]. To prove security, one relies on the non-standard one-more variant of the underlying assumption (e.g., one-more CDH for BLS).

Idealized Models. In addition to the works in the standard and random oracle model mentioned before, there are also constructions [FPS20, KLX22, TZ22] that are proven secure in more idealized models,

such as the algebraic or generic group model [FKL18, Sho97]. While it leads to efficient schemes, we want to avoid using such a model.

3.1.5 Subsequent Work

Here, we discuss results on blind signatures that have been published subsequent to our work. Katsumata, Reichle, and Sakai [KRS23] constructed a round-optimal and practical blind signature scheme in the pairing setting. Their construction is a careful instantiation of Fischlin's framework [Fis06]. Both communication and signature size are smaller than in our work, at the cost of a decisional assumption and the use of rewinding. In a more recent follow-up, Kastner, Nguyen, and Reichle [KNR23] again instantiate Fischlin's framework carefully using the strong RSA assumption to achieve efficient blind signatures without pairings.

Closely related to our work is the very recent achievement by Chairattana-Apirom, Tessaro, and Zhu [CATZ23]. They manage to construct efficient and secure pairing-free blind signatures in the random oracle model, based on various interactive and non-interactive variants of CDH. Interestingly, their construction from standard, non-interactive CDH builds on the ideas we introduced in our Rai-Choo construction (Section 3.7). Asymptotically, the efficiency of this pairing-free construction is comparable with the efficiency of Rai-Choo. In terms of concrete parameters, the avoidance of pairings results in smaller group elements and therefore in smaller signatures and communication. On the downside, their construction only achieves a weaker non-standard variant of one-more unforgeability, which can still be useful in various applications. Extending this result to full one-more unforgeability is an interesting open problem.

Two independent works [DHP23, KLR23] show attacks breaking concurrent security of blind signatures based on parallel repetition of an underlying identification scheme with small challenge space. In particular, this applies to schemes in the post-quantum setting, e.g., CSI-Otter [KLLQ23].

3.1.6 Outline

The rest of this chapter is structured as follows. We first give a detailed technical overview in Section 3.2. In Section 3.3, we give preliminaries that are only used in this chapter, including the definition of blind signatures. In Sections 3.4 to 3.6, we formally present our results from [CAHL+22a], namely, PI-Cut-Choo from CDH, the RSA-based construction, and a transformation from semi-honest signer blindness to malicious signer blindness. Finally, in Section 3.7, we present our CDH-based construction with stateless Signer from [HLW23a].

3.2 Technical Overview

In this section, we informally introduce the main ideas developed in this chapter. For that, we first recall the main idea of the boosting transform [KLR21]. Then, we present the ideas behind our results [CAHL+22a, HLW23a].

3.2.1 The Boosting Transform

We start this overview by recalling the boosting transform [KLR21]. Let BS be a blind signature scheme which is secure against an adversary that queries the Signer for a small number of signatures, namely, a logarithmic number in the security parameter. Specifically, Katz, Loss, and Rosenberg [KLR21] assume that BS is a linear blind signature scheme [HKL19]. For such a scheme, the signing protocol consists of three messages, namely, the commitment R, the challenge c, and the response s. The boosting transform results in a new scheme which is secure for any number of signing interactions between Signer and adversary. In the Nth signing interaction, the User and the Signer behave as follows:

1. The Signer sends N. The User commits to its message m using randomness $\varphi_j,\ j \in [N]$, thereby obtaining N commitments μ_j. It also samples random coins $\rho_j,\ j \in [N]$ for the user algorithm of BS. Then, it commits to each pair (μ_j, ρ_j) using a random oracle, and sends the resulting commitments com_j to the Signer.

2. Signer and User run the underlying scheme BS N times in parallel. We refer to these N parallel runs as *sessions*. More precisely, the Signer uses its secret key sk, and the User uses the public key pk, μ_i as the message, and ρ_j as the random coins in the jth session, for $j \in [N]$.

3. Before the final messages $s_j,\ j \in [N]$ are sent from the Signer to the User, the Signer selects a random session $J \in [N]$. The User now has to open all the commitments com_j for $j \in [N] \setminus \{J\}$ by sending (μ_j, ρ_j). The Signer can now verify that the User behaved honestly for all but the Jth session. In case the User behaved dishonestly in one session, the Signer aborts.

4. The Signer completes the Jth session by sending the final message s_J. Because of the structure of BS, this is the only point where the Signer needs sk. Finally, the User derives a signature σ_J from that session as in BS, and outputs $\sigma = (\sigma_J, \varphi_J)$ as its final signature.

Katz, Loss, and Rosenberg [KLR21] show that the above scheme is secure for polynomially many signing interactions, given that the underlying scheme BS is secure for logarithmically many signing interactions. In more detail, they provide a reduction that simulates a signer oracle for the new scheme, given a logarithmic number of queries to the signer oracle for BS. Their reduction distinguishes the following cases for the Nth signing interaction:

1. If the adversary (i.e., the User) is dishonest in at least two sessions, then the adversary is caught. Hence, no response has to be provided and no secret key is needed.

2. If the adversary is honest in all sessions, the reduction can extract all (μ_j, ρ_j) by inspecting random oracle queries. Using a special property of the underlying scheme BS, this allows the reduction to simulate the response by programming the random oracle.

3. If the adversary is dishonest in exactly one session j^*, then either $J \neq j^*$ and the adversary is caught, or $J = j^*$, and the reduction has to use the signer oracle of BS to provide the response s_J. In this case, we say that there is a *successful cheat*.

It is clear that the probability of a successful cheat is at most $1/N$ in the Nth signing interaction. Therefore, the expected number of successful cheats over q signing interactions is at most $\sum_{N=2}^{q+1} 1/N \leq O(\log q)$. Using an appropriate Chernoff bound, it therefore can be argued that the underlying signer oracle for BS is called logarithmically many times. Unfortunately, the transform yields impractical parameter sizes for the resulting signature scheme, which results from a relatively loose reduction to BS. In addition, the communication and computation now scales linearly with N:

a) In the second message, the User sends N commitments com_j.

b) In the third message, the Signer sends N commitments R_j.

c) In the fourth message, the User sends N challenges c_j.

d) In the sixth message, the User opens $N - 1$ of the commitments com_j.

Our first goal will be to eliminate these linear dependencies.

3.2.2 Efficient Boosting and PI-Cut-Choo

We now describe the main ideas of our first contribution [CAHL+22a].

Communication-Efficient Boosting. We first describe how we can generically avoid the linear dependencies a) to d) mentioned above[2]. First, we eliminate the linear dependency a) by replacing the commitments com_j by a single commitment com, which commits to (salted) hashes of all (μ_j, ρ_j) at once. By sending all (μ_j, ρ_j) for $j \neq J$ and the hash of (μ_J, ρ_J), the User can still open this commitment without revealing (μ_J, ρ_J). Next, we focus on d). Here, we let the User generate the randomness (ρ_j, φ_j) used for each session using the puncturable pseudorandom function [SW14][3]. We also replace the unstructured commitment with a commitment scheme that is homomorphic in its randomness. This allows the User to derive the commitments μ_i as rerandomizations $\mathsf{Com}(\mathsf{m}, \varphi_0 + \varphi_j)$ of one single commitment $\mu_0 = \mathsf{Com}(\mathsf{m}, \varphi_0)$ with randomness φ_j. The User would now initially send commitment μ_0 together with com. Later, the User can open the commitment com by sending only a punctured key k_J. The Signer can now recompute all (φ_j, ρ_j) using k_J and derive all μ_i from μ_0 using φ_j, for $j \neq J$. Intuitively, this preserves blindness, as the punctured key does not reveal anything about the randomness ρ_J, φ_J. Using similar tricks, we eliminate c). To tackle b) in the generic case, we compute the N values R_i of the underlying linear scheme BS as subset sums of a logarithmic number of such values. Then, only these basis values have to be sent.

We end up with a scheme with logarithmic communication complexity, for which the ideas that underlie the original boosting transform still apply. However, the scheme still has two main drawbacks that we will eliminate in our concrete constructions: first, we rely on the very loose security bound of the underlying linear blind signature scheme BS. The transform requires that BS supports a logarithmic but non-trivial number of signatures, which results in inefficient parameter requirements. Second, due to our solution of b), the logarithmic term of the communication complexity depends on computational assumptions. Thus, the loose bound will have a significant impact on communication complexity.

Parallel Instance Cut-And-Choose. Let us now explain how we solve this issues, first focusing on our pairing-based construction from CDH. We observe that by letting the cut-and-choose parameter grow slightly faster than before and scaling appropriately, the expected number of successful cheats can be bounded to be less than 1. Unfortunately, we can not just use the Chernoff bound if we want to argue that this also holds with overwhelming probability. We can, however, use the Chernoff bound to show that a single cheat happens with some constant probability less than 1. Then, we play our next card, which is parallel repetition. Namely, we run K independent instances of our scheme so far, where each instance is relative to a separate key pair. We show that with high probability, in one randomly chosen instance, there is *no cheat at all.* Using this observation, we can give a reduction from the key-only security of the underlying blind signature scheme, i.e., the base scheme BS does not have to support any signing query. We do not apply this overall strategy to a linear blind signature scheme, but instead to the BLS blind signature scheme [Bol03]. We notice that the approach also works for this scheme and observe additional benefits. First, the BLS scheme allows to aggregate signatures for a significant efficiency improvement. Second, the scheme has two rounds and thus the logarithmic term in the communication complexity is independent of computational assumptions, namely, dependency b) disappears. We emphasize that the original BLS blind signature scheme is secure under a one-more variant of the CDH assumption. Fortunately, we only need key-only security here, which is implied by CDH. Also, the concrete security loss of our scheme is as for the standard BLS digital signature scheme [BLS01], which means that it can be used over the same groups as BLS. To further improve concrete efficiency, we also introduce minor optimizations such as making the Signer commit to its cut-and-choose indices in its first message. In this way, the reduction in the blindness proof can extract these indices rather than guessing them. This leads to more efficient statistical security parameters[4].

Our Scheme from RSA. In the RSA regime, our construction is based on the Okamoto-Guillou-

[2]The generic communication-efficient boosting transform is part of [CAHL+22a] but not of this dissertation. This is because [CAHL+22a] is a merge of [WHL22] and [CAL22] and the generic transform was originally only present in [CAL22]. Still, most ideas of this generic transform are also present in our concrete schemes presented here and resulting from [WHL22].

[3]We instantiate the puncturable pseudorandom function efficiently using random oracles [GGM84].

[4]Note that without this optimization, the security loss would be exponential in K.

Quisquater (OGQ) [Oka93] linear function. Using this function, one can instantiate the boosting transform [KLR21] and our communication-efficient variant. Roughly, the function is parameterized by an RSA modulus n and a prime γ with $\gcd(n, \gamma) = \gcd(\varphi(n), \gamma) = 1$. As we can not aggregate signatures efficiently, we can not mimic the K-repetition technique from our CDH-based scheme. Hence, we need to find an alternative way to avoid the loose security bound and improve the communication complexity. For the former challenge, when increasing γ, which can be done independently from the modulus n, we observe that the bound becomes acceptable. Although this improves the bound and thus concrete parameters, we still have a rather large communication complexity, due to the logarithmic number of $R_i \in \mathbb{Z}_n^*$ that are sent in our generic transformation. Here, our solution is to send a short random seed and derive the values R_i using a random oracle. Now, the Signer has to recover the preimages of the R_i to continue the protocol. We show that the OGQ linear function admits a trapdoor, that allows to sample preimages, solving this problem as well.

3.2.3 PI-Cut-Choo Evolves to Rai-Choo

Here, we give an informal overview of our second contribution [HLW23a] in the realm of blind signatures. We assume familiarity with the previous parts of this technical overview. Reflecting on prior constructions [KLR21, CAHL+22a], observe the importance of growing the parameter N as a function of the number of signing interactions over time: it allows to bound the expected number of successful cheats. Thus, keeping N fixed presents several technical challenges that we discuss next.

Strawman One: Fixed Cut-and-Choose. As explained above, the key idea of PI-Cut-Choo is to ensure that for one of the parallel instances i^*, the adversary *never cheats* in any of its interactions with the Signer. This argument fails if we set N to be constant, e.g., $N = 2$. However, by keeping the number of parallel instances K the same, we can still argue that with overwhelming probability *in each signing interaction*, there is a non-cheating instance i^*. We highlight the reversed role of quantifiers: the non-cheating instance i^* could now be different for every signing interaction. Unfortunately, the reduction approach presented in PI-Cut-Choo only allows to embed the target public key of the underlying scheme BS in a *fixed key* among the keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ corresponding to the K parallel instances. Once this key is fixed, the reduction fails if ever there is a successful cheat with respect to this instance.

Strawman Two: Dynamic Key Structure (Naively). The above discussion shows that we have to support a *dynamic embedding* of the target public key into one of the keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$. The first naive idea would be to use a fresh set of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_K$ in each interaction. In PI-Cut-Choo, the base scheme BS is a two-move scheme, in which the first message c (challenge) sent from User to Signer does not depend on the public key. Thus, our reduction for the resulting scheme can identify the non-cheating instance i^* *after* seeing the User's commitments and challenges. Using this observation, we could let the Signer send the fresh public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ that will be used in the current signing interaction after receiving commitments and challenges. This way, the reduction knows in which key pk_{i^*} to embed the target public key in each signing interaction. To do so, the reduction first identifies the non-cheating instance $i^* \in [K]$, and then samples $(\mathsf{pk}_i, \mathsf{sk}_i)$ for $i \neq i^*$ honestly, while setting pk_{i^*} to (a rerandomization of) the target public key. Finally, the reduction can use sk_i to simulate all instances except i^*, while using random oracle programming in the non-cheating instance i^*. We can use random-self reducibility of the underlying signature scheme to ensure blindness of this construction. Namely, the User rerandomizes the keys and signatures it receives from the Signer. Otherwise, it would be trivial to link signatures to signing interactions. The final signature then contains the rerandomized set of keys and signatures. Fortunately, the BLS scheme [Bol03], which serves as the basis of PI-Cut-Choo, has such a property.

However, the above scheme is insecure. Since a fresh set of keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ is used in every interaction, there is nothing tying signatures to the Signer's actual public and secret key. In particular, there is no way from preventing the adversary from trivially creating a forgery containing a set of keys

of its own choice. In the security proof, the reduction can not extract a forgery for BS with respect to the target public key in this scenario.

Our Solution: PI-Cut-Choo evolves to Rai-Choo. To overcome the remaining issues of the above strawman approach, we fix one public key pk and one secret key sk for our scheme. Instead of using independent public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ for each interaction, we instead use a sharing

$$(\mathsf{pk}_1, \mathsf{sk}_1), \ldots, (\mathsf{pk}_K, \mathsf{sk}_K) \text{ such that } \sum_i \mathsf{sk}_i = \mathsf{sk} \text{ and } \prod_i \mathsf{pk}_i = \mathsf{pk}.$$

By setting pk to be the target public key of the underlying scheme BS and carefully working out the details, our reduction is now able to extract a forgery as required. It remains to sketch why the simulation of the signing oracle is still possible with this new structure of the $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$. Note that the reduction can define the $\mathsf{pk}_1, \ldots, \mathsf{pk}_K$ in a way that allows it to know all but one sk_i. Concretely, after identifying the non-cheating instance i^* in an interaction with the adversary, the reduction first samples $(\mathsf{pk}_i, \mathsf{sk}_i)$ for all $i \in [K] \setminus \{i^*\}$, and then sets $\mathsf{pk}_{i^*} := \mathsf{pk} \cdot \prod_{i \neq i^*} \mathsf{pk}_i^{-1}$. This is identically distributed to the real sharing.

In summary, we have successfully transformed a key-only secure scheme BS into a fully secure one, while using a constant cut-and-choose parameter N. We can further optimize the scheme using many minor tricks, some of them similar to [CAHL+22a]. In the process we also manage to reduce the number of moves to two, which is optimal. This is because in our new scheme, we can make the cut-and-choose step completely non-interactive using a random oracle, and the Signer does not need to send N anymore, as it is fixed.

3.3 Preliminaries for this Chapter

In addition to the general preliminaries in Chapter 2, we introduce preliminaries that are only relevant for this chapter. This includes, most importantly, the definition of blind signatures.

Chernoff Bound. In our proof, we make use of a specific Chernoff bound following [KLR21]. For completeness, we state and prove this bound below.

Lemma 3.1. *For a sum X of independent random variables over $\{0,1\}$ and any $s > \mathbb{E}[X]$, we have $\Pr[X \geq s] \leq \exp(3\mathbb{E}[X] - s)$, where $\exp(x) := e^x$.*

Proof. The proof is similar to [KLR21]. Recall the standard Chernoff bound for all $\delta > 0$:

$$\Pr[X \geq (1+\delta) \cdot \mathbb{E}[X]] \leq \exp\left(-\mathbb{E}[X] \frac{\delta^2}{2+\delta}\right).$$

Using $x^2 > (x+2)(x-2)$ for all $x \geq 0$ we obtain

$$\begin{aligned}
\Pr[X \geq s] &= \Pr\left[X \geq \left(1 + \left(\frac{s}{\mathbb{E}[X]} - 1\right)\right) \cdot \mathbb{E}[X]\right] \\
&\leq \exp\left(-\mathbb{E}[X] \frac{(s/\mathbb{E}[X] - 1)^2}{2 + (s/\mathbb{E}[X] - 1)}\right) \\
&\leq \exp\left(-\mathbb{E}[X] \left(\frac{s}{\mathbb{E}[X]} - 3\right)\right) = \exp(3\mathbb{E}[X] - s).
\end{aligned}$$

□

Puncturable Pseudorandom Functions. We recall the definition of puncturable pseudorandom functions, following Sahai and Waters [SW14].

Definition 3.1 (Puncturable Pseudorandom Function)**.** *A puncturable pseudorandom function (PPRF) is a triple of PPT algorithms* $\mathsf{PRF} = (\mathsf{Gen}, \mathsf{Puncture}, \mathsf{Eval})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, 1^{d(\lambda)}) \to k$ *takes as input* 1^λ *and an input length* $1^{d(\lambda)}$*, and outputs a key* k*.*
- $\mathsf{Puncture}(k, X) \to k_X$ *takes as input a key* k *and a polynomial size set* $\emptyset \neq X \subseteq \{0,1\}^{d(\lambda)}$*, and outputs a punctured key* k_X*.*
- $\mathsf{Eval}(k, x) \to r$ *is deterministic, takes a key* k *and an element* $x \in \mathcal{D}$ *as input, and outputs an element* $r \in \{0,1\}^{r(\lambda)}$*.*

Further, the following completeness and security properties should hold:

- ***Completeness of Puncturing.*** *For any* $d(\lambda) = \mathsf{poly}(\lambda)$*, any* $X \subseteq \{0,1\}^{d(\lambda)}$*, any* $k \in \mathsf{Gen}(1^\lambda, 1^{d(\lambda)})$*, any* $k_X \in \mathsf{Puncture}(k, X)$*, and any* $x' \notin X$ *we have* $\mathsf{Eval}(k, x') = \mathsf{Eval}(k_X, x')$*.*
- ***Pseudorandomness.*** *For any* $d(\lambda) = \mathsf{poly}(\lambda)$ *and any PPT algorithm* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{A},\mathsf{PRF},d}(\lambda) := \left| \Pr\left[\mathcal{A}(St, k_X, (r_x)_{x\in X}) = 1 \;\middle|\; \begin{array}{l} (X, St) \leftarrow \mathcal{A}(1^\lambda), \\ k \leftarrow \mathsf{Gen}(1^\lambda, 1^{d(\lambda)}), \\ k_X \leftarrow \mathsf{Puncture}(k, X), \\ r_x := \mathsf{Eval}(k, x) \text{ for } x \in X \end{array} \right] \right.$$
$$\left. - \Pr\left[\mathcal{A}(St, k_X, (r_x)_{x\in X}) = 1 \;\middle|\; \begin{array}{l} (X, St) \leftarrow \mathcal{A}(1^\lambda), \\ k \leftarrow \mathsf{Gen}(1^\lambda, 1^{d(\lambda)}), \\ k_X \leftarrow \mathsf{Puncture}(k, X), \\ r_x \xleftarrow{\$} \{0,1\}^{r(\lambda)} \text{ for } x \in X \end{array} \right] \right|.$$

Puncturable pseudorandom function can easily be constructed using random oracles. In the following, we recall the well-known GGM construction [GGM84]. Let $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a random oracle. For simplicity, we write $\mathsf{H}(x) = (\mathsf{H}_0(x), \mathsf{H}_1(x))$ for any x to split the output of H into two λ-bit strings. Keys are random strings of length λ and for $\ell \in \mathbb{N}, x \in \{0,1\}^\ell, k \in \{0,1\}^\lambda$, we define $\mathsf{GGM}_{0,k}() := k$ and recursively $\mathsf{GGM}_{\ell,k}(b \,\|\, x) := \mathsf{GGM}_{\ell-1,\mathsf{H}_b(k)}(x)$. Then, the evaluation of the pseudorandom function with key $k \in \{0,1\}^\lambda$ on input $x \in \{0,1\}^{d(\lambda)}$ is $\mathsf{PRF.Eval}(k, x) := \mathsf{GGM}_{d(\lambda),k}(x)$. We set $\mathsf{Puncture}_0(k, X) := \emptyset$ and define an algorithm $\mathsf{Puncture}_\ell(k, X)$ to puncture keys at a set of points $\emptyset \neq X \subseteq \{0,1\}^\ell$ as follows:

- Set $k_X := \emptyset$ and $(k_0, k_1) := \mathsf{H}(k)$.
- Define sets $X_b := \{x \mid (x_1, x) \in X \wedge x_1 = b\}$ for $b \in \{0,1\}$.
- If $X_0 = \emptyset$, set $k_X := k_X \cup \{k_0\}$. Else set $k_X := k_X \cup \{\mathsf{Puncture}_{\ell-1}(k_0, X_0)\}$.
- If $X_1 = \emptyset$, set $k_X := k_X \cup \{k_1\}$. Else set $k_X := k_X \cup \{\mathsf{Puncture}_{\ell-1}(k_1, X_1)\}$.
- Return k_X.

This algorithm always terminates. We set $\mathsf{PRF.Puncture}(k, X) := \mathsf{Puncture}_{d(\lambda)}(k, X)$. Note that a punctured key contains all information needed to evaluate the pseudorandom function at inputs that are not in X. Using induction over $d(\lambda)$ and $|X|$, one can easily show that the number of elements in k_X is at most $(d(\lambda) - 1)|X| + 1$. It remains to show pseudorandomness on punctured points.

Lemma 3.2. *Let* $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ *be a random oracle and consider the puncturable pseudorandom function* PRF *as defined above. Let* $\mathcal{A}$ *be an algorithm that makes at most* Q *queries to* H*. Then, for any* $d(\lambda) = \mathsf{poly}(\lambda)$*, we have Then the advantage of* $\mathcal{A}$ *in the pseudorandomness game for* PRF *is at most*

$$\mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{A},\mathsf{PRF},d}(\lambda) \leq \frac{(2d(\lambda) - 1)Q|X|}{2^\lambda},$$

where X *is the set of points that* $\mathcal{A}$ *outputs.*

Proof. A simple hybrid argument shows that we only have to argue that we have pseudorandomness for keys which are punctured at one point. Then we can show the claim by induction over the input length $d = d(\lambda)$. In particular, we show that for any PPT algorithm making at most Q random oracle queries the advantage can be bounded by $(2d-1)Q/2^\lambda$. We start with the case of $d = 1$. Let $\mathcal{A}$ be a PPT algorithm and assume it outputs $X \subseteq \{0,1\}, |X| = 1$. Let $k \xleftarrow{\$} \{0,1\}^\lambda$ be a random key. Let $X = \{x\}$. Conditioned on $k_X = \{\mathsf{H}_{1-x}(k)\}$ the value $\mathsf{PRF.Eval}(k,x) = \mathsf{H}_x(k)$ is uniformly random unless $\mathcal{A}$ queries $\mathsf{H}(k)$. As k is sampled uniformly at random and $\mathcal{A}$ can only make a polynomial number of random oracle queries, a union bound shows that the probability that this happens is negligible. In more detail the distinguishing advantage can be upper bounded by $Q/2^\lambda$. Now consider $d > 1$, let $k \xleftarrow{\$} \{0,1\}^\lambda$ be a random key and let $X = \{x\}, x \in \{0,1\}^d$ be $\mathcal{A}$'s initial output. Let k_X, r be the values that $\mathcal{A}$ gets as input after outputting X. Write $x = x_1 \parallel \bar{x}$ for $x_1 \in \{0,1\}, \bar{x} \in \{0,1\}^{d-1}$. We show indistinguishability via a sequence of four games. In the first game, we let k_X be the honestly punctured key $k_X = \{s_{1-x_1} = \mathsf{H}_{1-x_1}(k), k_{\{\bar{x}\}}\}$ and $r = \mathsf{Eval}(k,x)$ be the real evaluation at input x. In the second game, we set $s_{1-x_1} \xleftarrow{\$} \{0,1\}^\lambda$. Note that similarly to the argument for $d = 1$, the adversary $\mathcal{A}$ can only distinguish between these two games, if it queries $\mathsf{H}(k)$, which happens with probability at most $Q/2^\lambda$. In the third game, we sample $r \xleftarrow{\$} \{0,1\}^\lambda$. Note that any distinguisher between the second and the third game can be turned into a distinguisher for input length $d-1$ with the same advantage by a straight forward reduction. Hence, using the induction hypothesis, the advantage of $\mathcal{A}$ in distinguishing the second and the third game can be upper bounded by $(2(d-1)-1)Q/2^\lambda$. Finally, we undo the change we did in the second game. That is, we set $s_{1-x_1} = \mathsf{H}_{1-x_1}(k)$. Again, the advantage of distinguishing between the third and fourth game is at most $Q/2^\lambda$. In total, we obtain that the advantage of $\mathcal{A}$ in distinguishing between the real value of the pseudorandom function at input x and a random string is at most $(2d-1)Q/2^\lambda$. □

Randomness Homomorphic Commitments. We define a special type of perfectly hiding commitment scheme in which the randomness can be rerandomized publicly.

Definition 3.2 (Randomness Homomorphic Commitment Scheme)**.** *A randomness homomorphic commitment scheme is a tuple of PPT algorithms* $\mathsf{CMT} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Translate})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{ck}$ *takes as input the security parameter* 1^λ *and outputs a commitment key* ck*. We assume that* ck *implicitly defines a message space* $\mathcal{M}_{\mathsf{ck}}$ *and a randomness space* $\mathcal{R}_{\mathsf{ck}}$*. Further, we assume that* $\mathcal{R}_{\mathsf{ck}}$ *is a group with respect to an efficiently computable group operation* $+$*.*
- $\mathsf{Com}(\mathsf{ck}, x; r) \to \mu$ *takes as input a key* ck*, an element* $x \in \mathcal{M}_{\mathsf{ck}}$*, a randomness* $r \in \mathcal{R}_{\mathsf{ck}}$ *and outputs a commitment* $\mu \in \{0,1\}^*$*.*
- $\mathsf{Translate}(\mathsf{ck}, \mu, r) \to \mu'$ *is deterministic, takes a key* ck*, a commitment* $\mu \in \{0,1\}^*$*, and a randomness* $r \in \mathcal{R}_{\mathsf{ck}}$ *as input and outputs a commitment* μ'*.*

Further, the following security and completeness properties should hold:

- ***Completeness of Translation.*** *For any* $\mathsf{ck} \in \mathsf{Gen}(1^\lambda)$*, any* $x \in \mathcal{M}_{\mathsf{ck}}$*, and any* $r, r' \in \mathcal{R}_{\mathsf{ck}}$*, we have*
$$\mathsf{Translate}(\mathsf{ck}, \mathsf{Com}(\mathsf{ck}, x; r), r') = \mathsf{Com}(\mathsf{ck}, x; r + r').$$
- ***Perfectly Hiding.*** *For any key* ck *and any* $x_0, x_1 \in \mathcal{M}_{\mathsf{ck}}$*, the following distributions are identical:*
$$\left\{ (\mathsf{ck}, x_0, x_1, \mu) \;\middle|\; \begin{array}{l} r \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}, \\ \mu := \mathsf{Com}(\mathsf{ck}, x_0; r) \end{array} \right\} \text{ and } \left\{ (\mathsf{ck}, x_0, x_1, \mu) \;\middle|\; \begin{array}{l} r \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}, \\ \mu := \mathsf{Com}(\mathsf{ck}, x_1; r) \end{array} \right\}.$$
- ***Computationally Binding.*** *For any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*
$$\mathsf{Adv}^{\mathsf{bind}}_{\mathcal{A},\mathsf{CMT}}(\lambda) := \Pr\left[\begin{array}{ll} & \mathsf{Com}(\mathsf{ck}, x_0; r_0) = \mathsf{Com}(\mathsf{ck}, x_1; r_1) \\ \wedge & x_0 \neq x_1 \end{array} \;\middle|\; \begin{array}{l} \mathsf{ck} \leftarrow \mathsf{Gen}(1^\lambda), \\ (x_0, r_0, x_1, r_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array} \right].$$

Randomness homomorphic commitment schemes can easily be obtained from standard assumptions, such as RSA and DLOG. As a first example, a folklore RSA-based commitment scheme obtained from the Guillou-Quisquater identification scheme [GQ90] is randomness homomorphic. The commitment key ck contains public RSA parameters (n, e) such that $n = pq$ for two distinct large primes, e is prime, and $\gcd(e, \varphi(n)) = 1$. The key ck also contains $y := x^e \bmod n$, where $x \xleftarrow{\$} \mathbb{Z}_n^*$. Commitment and translation algorithms for $\mathsf{m} \in \mathbb{Z}_e, r \in \mathbb{Z}_n^*, \mu \in \mathbb{Z}_n^*$ are defined as

$$\mathsf{Com}(\mathsf{ck}, \mathsf{m}; r) := r^e y^{\mathsf{m}} \bmod n, \quad \mathsf{Translate}(\mathsf{ck}, \mu, r) := r^e \mu \bmod n.$$

It is easy to observe that translation is complete and the commitment is perfectly hiding. To see that it is computationally binding, note that given two pairs $(\mathsf{m}_0, r_0), (\mathsf{m}_1, r_1) \in \mathbb{Z}_e \times \mathbb{Z}_n^*$ with $\mathsf{m}_0 \neq \mathsf{m}_1$ and $\mathsf{Com}(\mathsf{ck}, \mathsf{m}_0; r_0) = \mathsf{Com}(\mathsf{ck}, \mathsf{m}_1; r_1)$ we have $r_0^e y^{\mathsf{m}_0} \equiv r_1^e y^{\mathsf{m}_1} \pmod n$. Without loss of generality we have $\mathsf{m}_0 > \mathsf{m}_1$ and as e is prime we have $\gcd(e, \mathsf{m}_0 - \mathsf{m}_1) = 1$. Thus, we can apply Shamir's trick to $(r_0^{-1} r_1)^e \equiv y^{\mathsf{m}_0 - \mathsf{m}_1} \pmod n$ and derive an e^{th} root of y, solving RSA.

In the DLOG setting, the standard Pedersen commitment scheme [Ped92] is randomness homomorphic, where the commitment key is a pair of group elements g, h, where $(\mathbb{G}, g, p) \leftarrow \mathsf{GGen}(1^\lambda)$. Commitment and translation algorithms for $\mathsf{m} \in \mathbb{Z}_p, r \in \mathbb{Z}_p, \mu \in \mathbb{G}$ are defined as

$$\mathsf{Com}(\mathsf{ck}, \mathsf{m}; r) := g^r h^{\mathsf{m}}, \quad \mathsf{Translate}(\mathsf{ck}, \mu, r) := g^r \mu.$$

It is well-known and easy to see that the scheme is perfectly hiding and computationally binding under the DLOG assumption relative to GGen. Also, completeness of translation is easy to see.

Non-Interactive Proof Systems. For our generic transformation from semi-honest signer blindness to malicious signer blindness, we need non-interactive proof systems. For simplicity of exposition, we focus on online-extractable proof systems in the random oracle model as defined and constructed in [Fis05]. However, any reasonable notion of non-interactive proof system with zero-knowledge and proof-of-knowledge property is applicable in our context. Let $\mathcal{R} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. We define the associated language $\mathcal{L}_\mathcal{R} \subseteq \{0,1\}^*$ to contain all $x \in \{0,1\}^*$ such that there is an $w \in \{0,1\}^*$ with $(x, w) \in \mathcal{R}$. Here, we call x the statement and w the witness. In the following, we focus on **NP** relations, which means that $\mathcal{R}$ is efficiently decidable and the length of the witness w is bounded by a polynomial in the length of the statement x. Relations are allowed to implicitly depend on the security parameter, with the restriction that the length of statments is polynomially bounded in the security parameter.

Definition 3.3 (Non-Interactive Proofs)**.** *Let $\mathcal{R}$ be an* **NP** *relation and* H *be a random oracle. A non-interactive zero-knowledge proof-of-knowledge (NIZKPOK) for $\mathcal{R}$ is a tuple* $\mathsf{PS} = (\mathsf{PProve}, \mathsf{PVer})$ *of PPT algorithms such that*

- $\mathsf{PProve}^{\mathsf{H}}(x, w) \to \pi$ *takes as input a statement x and a witness w, and outputs a proof π.*
- $\mathsf{PVer}^{\mathsf{H}}(x, \pi) \to b$ *is deterministic, takes as input x and a proof π, and outputs a bit $b \in \{0,1\}$.*

Further, the following completeness and security properties should hold:

- ***Completeness.*** *For all $(x, w) \in \mathcal{R}$ and all $\pi \in \mathsf{PProve}^{\mathsf{H}}(x, w)$, we have $\mathsf{PVer}^{\mathsf{H}}(x, \pi) = 1$.*
- ***Zero-Knowledge.*** *There exists a PPT algorithm* Sim *such that for any PPT algorithm $\mathcal{D}$ the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{D},\mathsf{PS},\mathsf{Sim}}(\lambda) := \left| \Pr\left[\mathcal{D}^{\mathsf{H}}(St, \pi) = 1 \;\middle|\; \begin{array}{l} (St, x, w) \leftarrow \mathcal{D}^{\mathsf{H}}(1^\lambda), \\ \pi \leftarrow \mathsf{PProve}^{\mathsf{H}}(x, w) \end{array} \right] - \Pr\left[\mathcal{D}^{\mathsf{H}}(St, \pi) = 1 \;\middle|\; \begin{array}{l} (St_s, \mathsf{H}_0) \leftarrow \mathsf{Sim}(1^\lambda), \\ (St, x, w) \leftarrow \mathcal{D}^{\mathsf{H}_0}(1^\lambda), (\mathsf{H}, \pi) \leftarrow \mathsf{Sim}(St_s, x) \end{array} \right] \right|.$$

- ***Proof-of-Knowledge.*** *There exists a PPT algorithm* Ext *such that for any algorithm* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{pok}}_{\mathcal{A},\mathsf{PS},\mathsf{Ext}}(\lambda) := \Pr\left[(x,w) \notin \mathcal{R} \wedge \mathsf{PVer}^{\mathsf{H}}(x,\pi) = 1 \;\middle|\; \begin{array}{l}(x,\pi) \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda),\\ w \leftarrow \mathsf{Ext}(x,\pi,\mathcal{Q})\end{array}\right],$$

where $\mathcal{Q}$ *denotes the list of queries of* $\mathcal{A}$ *to oracle* H *and the respective hash values.*

Blind Signatures. Blind signatures are the main primitive of interest for this chapter. In a blind signature scheme, we consider a Signer that generates his key material, namely public key pk and secret key sk using an algorithm Gen. The Signer can then interact with users to create signatures. Precisely, the Signer holds the secret key sk and the User holds the public key pk and a message m to be signed. To create a signature, they engage in an interactive protocol, formally specified by interactive algorithms S and U. When the protocol is completed, algorithm U outputs a signature σ, which can then be verified by an algorithm Ver with respect to the message m and the public key pk.

Definition 3.4 (Blind Signature Scheme)**.** *A blind signature scheme is a quadruple of PPT algorithms* $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input the security parameter* 1^λ *and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*. We assume that the public key* pk *defines a message space* $\mathcal{M} = \mathcal{M}_{\mathsf{pk}}$ *implicitly.*
- S *and* U *are interactive algorithms, where* S *takes as input a secret key* sk *and* U *takes as input a key* pk *and a message* $\mathsf{m} \in \mathcal{M}$*. After the execution,* U *returns a signature* σ *and we write* $(\bot, \sigma) \leftarrow \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, \mathsf{m})\rangle$*.*
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$ *is deterministic and takes as input public key* pk*, message* $\mathsf{m} \in \mathcal{M}$*, and a signature* σ*, and returns* $b \in \{0,1\}$*.*

We require that BS *is complete in the following sense. For all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$ *and all* $\mathsf{m} \in \mathcal{M}_{\mathsf{pk}}$ *it holds that*

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) = 1 \mid (\bot, \sigma) \leftarrow \langle \mathsf{S}(\mathsf{sk}), \mathsf{U}(\mathsf{pk}, \mathsf{m})\rangle\right] = 1.$$

One-more unforgeability intuitively guarantees that users need to interact with the Signer to obtain signatures. In other words, if an adversary completes at most $k-1$ interactions with the Signer, then it should not be able to output signatures for k messages.

Definition 3.5 (One-More Unforgeability)**.** *Let* $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ *be a blind signature scheme and* $\ell \colon \mathbb{N} \to \mathbb{N}$*. For an algorithm* $\mathcal{A}$*, we consider the following game* ℓ-$\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda)$*:*

1. *Sample keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*.*
2. *Let* O *be an interactive oracle simulating* $\mathsf{S}(\mathsf{sk})$*. Run* $((\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)) \leftarrow \mathcal{A}^{O}(\mathsf{pk})$*, where* $\mathcal{A}$ *can query* O *in an arbitrarily interleaved way and complete at most* $\ell = \ell(\lambda)$ *of the interactions with* O*.*
3. *Output* 1 *if and only if all* $\mathsf{m}_i, i \in [k]$ *are distinct,* $\mathcal{A}$ *completed at most* $k-1$ *interactions with* O *and for each* $i \in [k]$ *it holds that* $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}_i, \sigma_i) = 1$*.*

We say that BS *is* ℓ*-one-more unforgeable (*ℓ-OMUF*), if for every PPT algorithm* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda) := \Pr\left[\ell\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda) \Rightarrow 1\right].$$

We say that BS *is one-more unforgeable (*OMUF*), if it is* ℓ-OMUF *for all polynomial* ℓ*.*

From a practical perspective, it is sufficient to focus on ℓ-OMUF for some large but a priori bounded ℓ, e.g., $\ell = 2^{30}$, while full OMUF is more of theoretical interest. Blindness protects the privacy of the User. Namely, it guarantees that the Signer can not link the pair (m, σ) to the signing interaction, and especially, it can not learn anything about the message m during the interaction.

Definition 3.6 (Blindness)**.** *Consider a blind signature scheme* $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$*. For an algorithm* $\mathcal{A}$ *and bit* $b \in \{0,1\}$*, consider the following game* $\mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$*:*

1. *Sample* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and run* $(\mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{sk})$*.*
2. *Let* O_0 *be an interactive oracle simulating* $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b)$ *and* O_1 *be an interactive oracle simulating* $\mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$*. Run* $\mathcal{A}$ *on input* St *with arbitrary interleaved one-time access to each of these oracles, i.e.,* $St' \leftarrow \mathcal{A}^{O_0, O_1}(St)$*.*
3. *Let* σ_b, σ_{1-b} *be the local outputs of* O_0, O_1*, respectively. If* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$*, then run* $b' \leftarrow \mathcal{A}(St', \bot, \bot)$*. Else, obtain a bit* b' *from* $\mathcal{A}$ *on input* σ_0, σ_1*, i.e., run* $b' \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$*.*
4. *Output* b'*.*

We say that BS *satisfies honest signer blindness, if for every PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) := \left| \Pr\left[\mathbf{BLIND}^{\mathcal{A}}_{0,\mathsf{BS}}(\lambda) \Rightarrow 1\right] - \Pr\left[\mathbf{BLIND}^{\mathcal{A}}_{1,\mathsf{BS}}(\lambda) \Rightarrow 1\right] \right|.$$

We also consider semi-honest and malicious signer blindness, where we modify the game in the following way:

- *For semi-honest signer blindness,* $(\mathsf{pk}, \mathsf{sk})$ *is not sampled by the game, but* $\mathcal{A}$ *outputs random coins* ρ *in addition to* $\mathsf{m}_0, \mathsf{m}_1$*. Then, the game defines* $(\mathsf{pk}, \mathsf{sk})$ *via* $(\mathsf{pk}, \mathsf{sk}) := \mathsf{Gen}(1^\lambda; \rho)$*.*
- *For malicious signer blindness,* $(\mathsf{pk}, \mathsf{sk})$ *is not sampled by the game, but* $\mathcal{A}$ *outputs* pk *in addition to* $\mathsf{m}_0, \mathsf{m}_1$*.*

Semi-honest signer blindness is a non-standard notion and lies inbetween honest and malicious signer blindness. We will show that any semi-honest signer blind scheme can be transformed into a malicious signer blind scheme while preserving one-more unforgeability. The high-level idea is to append a non-interactive zero-knowledge proof-of-knowledge to the public key. This proof shows that the Signer knows corresponding random coins that generate the key. The rest of the scheme does not change, and thus the transformation is very efficient.

Batched Partially Blind Signatures. We introduce a formal model for batched partially blind signatures. To recall, a partially blind signature scheme [AF96, AO00] allows to sign messages with respect to some public information string info, that the Signer knows. This string acts as a form of domain separator. Namely, one-more unforgeability for partially blind signatures guarantees that the User can output at most ℓ valid message signature pairs with respect to any public information string info, for which it interacted at most ℓ times with the Signer oracle. In the batched setting, we assume that the User holds multiple messages that should be signed in a single protocol run, which can reduce the amortized communication complexity. Batching has been subject of study for other primitives, e.g., in oblivious transfer [IKNP03, BBDP22].

Let us now introduce the syntax of batched partially blind signatures. Recall that in partially blind signatures, the Signer gets the public information string info, while the User gets info and the message m. Here, we generalize the syntax of partially blind signatures to the setting, where both the User and Signer get the batch size L as input, and multiple pairs $(\mathsf{info}_l, \mathsf{m}_l)$ are signed. This models that the batch size is not fixed, but instead it can be chosen dynamically.

Definition 3.7 (Batched Partially Blind Signature Scheme)**.** *A batched partially blind signature scheme is a quadruple of PPT algorithms* $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input the security parameter* 1^λ *and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*. We assume that the public key* pk *defines a message space* $\mathcal{M} = \mathcal{M}_{\mathsf{pk}}$*, and a public information space* $\mathcal{I} = \mathcal{I}_{\mathsf{pk}}$ *implicitly.*

- S *and* U *are interactive algorithms, where* S *takes as input a secret key* sk*, a batch size* $L \in \mathbb{N}$*, and* L *strings* $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$*, and* U *takes as input a key* pk*, a batch size* $L \in \mathbb{N}$*, and* L *pairs of messages* $\mathsf{m}_1, \ldots, \mathsf{m}_L \in \mathcal{M}$ *and strings* $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$*. After the execution,* U *returns* L *signatures* $\sigma_1, \ldots, \sigma_L$ *and we write*

$$(\bot, (\sigma_1, \ldots, \sigma_L)) \leftarrow \langle \mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]}), \mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l\in[L]}) \rangle.$$

- $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}, \mathsf{m}, \sigma) \to b$ *is deterministic and takes as input public key* pk*, a string* $\mathsf{info} \in \mathcal{I}$*, message* $\mathsf{m} \in \mathcal{M}$*, and a signature* σ*, and returns* $b \in \{0, 1\}$*.*

We require that BPBS *is complete in the following sense. For all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda)$*, all* $L = \mathsf{poly}(\lambda)$*, all* $\mathsf{m}_1, \ldots, \mathsf{m}_L \in \mathcal{M}_{\mathsf{pk}}$*, and all* $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}_{\mathsf{pk}}$ *it holds that*

$$\Pr\left[\forall l \in [L] : b_l = 1 \;\middle|\; \begin{array}{l} (\bot, (\sigma_1, \ldots, \sigma_L)) \\ \quad \leftarrow \langle \mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]}), \mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l\in[L]}) \rangle, \\ \forall l \in [L] : b_l := \mathsf{Ver}(\mathsf{pk}, \mathsf{info}_l, \mathsf{m}_l, \sigma_l) \end{array}\right] = 1.$$

In terms of security, we require the same security guarantees, as if we just run a normal (partially) blind signature scheme L times in parallel. We let the adversary determine the batch size in each interaction separately. This leads to a natural definition of batch one-more unforgeability.

Definition 3.8 (Batch One-More Unforgeability)**.** *Let* $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ *be a batched partially blind signature scheme and* $\ell \colon \mathbb{N} \to \mathbb{N}$*. For an algorithm* $\mathcal{A}$*, we consider the following game* ℓ-$\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BPBS}}(\lambda)$*:*

1. *Sample keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$*.*
2. *Let* O *be an interactive oracle, taking a batch size* $L \in \mathbb{N}$ *and* L *strings* $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$ *as input, and then simulating* $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]})$*. Run* $((\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)) \leftarrow \mathcal{A}^O(\mathsf{pk})$*, where* $\mathcal{A}$ *can query* O *in an arbitrarily interleaved way. Let* $\mathcal{C}$ *denote the list of all tuples* $(i, L, (\mathsf{info}_l)_{l\in[L]})$ *such that* $\mathcal{A}$ *submitted batch size* L *and strings* $(\mathsf{info}_l)_{l\in[L]}$ *in the* i*th completed interaction with* O*. It is required to hold that* $\sum_{(i,L,(\mathsf{info}_l)_{l\in[L]})\in\mathcal{C}} L \leq \ell$*.*
3. *For each* $\mathsf{info} \in \mathcal{I}$*, define the sets completed interactions and outputs*

$$\begin{aligned} \mathsf{Compl}[\mathsf{info}] &:= \{(i, l_0) \mid \exists (i, L, (\mathsf{info}_l)_{l\in[L]}) \in \mathcal{C} : \mathsf{info}_{l_0} = \mathsf{info}\} \\ \mathsf{Out}[\mathsf{info}] &:= \{i \in [k] \mid \mathsf{info}_i = \mathsf{info}\}. \end{aligned}$$

Output 1 *if and only if there is some* $\mathsf{info}^* \in \mathcal{I}$ *such that all* $\mathsf{m}_i, i \in \mathsf{Out}[\mathsf{info}^*]$ *are distinct,* $|\mathsf{Compl}[\mathsf{info}^*]| < |\mathsf{Out}[\mathsf{info}^*]|$*, and for each* $i \in \mathsf{Out}[\mathsf{info}^*]$ *it holds that* $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}_i, \mathsf{m}_i, \sigma_i) = 1$*.*

We say that BPBS *is* ℓ*-batch-one-more unforgeable (*ℓ-BOMUF*), if for every PPT algorithm* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\ell\text{-}\mathsf{BOMUF}}_{\mathcal{A},\mathsf{BPBS}}(\lambda) := \Pr\left[\ell\text{-}\mathbf{BOMUF}^{\mathcal{A}}_{\mathsf{BPBS}}(\lambda) \Rightarrow 1\right].$$

We say that BPBS *is batch one-more unforgeable (*BOMUF*), if it is* ℓ-BOMUF *for all polynomial* ℓ*.*

As for unforgeability, blindness should give the same guarantees as if we just run a normal (partially) blind signature scheme L times in parallel. Especially, it should not be possible to tell if two signatures result from the same interaction or not. In our security game, we let the malicious Signer choose two batches of (potentially different) sizes L_0 and L_1. The Signer also points to one element for each batch. Then, the game either swaps these two elements, or not, and the Signer has to distinguish these two cases. Via a hybrid argument, this implies that the Signer does not know which message is signed in which interaction.

Definition 3.9 (Batch Partial Blindness)**.** *Consider a batch partially blind signature scheme* $\mathsf{BPBS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$*. For an algorithm* $\mathcal{A}$ *and bit* $b \in \{0,1\}$*, consider the following game* $\mathbf{BBLIND}^{\mathcal{A}}_{b,\mathsf{BPBS}}(\lambda)$*:*

1. *Run* $\left(\mathsf{pk}, (\mathsf{info}_{l,0}, \mathsf{m}_{l,0})_{l\in[L_0]}, (\mathsf{info}_{l,1}, \mathsf{m}_{l,1})_{l\in[L_1]}, l_{*,0}, l_{*,1}, St\right) \leftarrow \mathcal{A}(1^\lambda)$*. Then, if* $\mathsf{info}_{l_{*,0},0} \neq \mathsf{info}_{l_{*,1},1}$*, then return* 0*.*
2. *If* $b = 1$*, then swap* $\mathsf{m}_{l_{*,0},0}$ *and* $\mathsf{m}_{l_{*,1},1}$*. That is, set* $\mathsf{m}' := \mathsf{m}_{l_{*,0},0}, \mathsf{m}_{l_{*,0},0} := \mathsf{m}_{l_{*,1},1}, \mathsf{m}_{l_{*,1},1} := \mathsf{m}'$*.*
3. *Let* O_0 *and* O_1 *be interactive oracles simulating*

$$\mathsf{U}(\mathsf{pk}, L_0, (\mathsf{m}_{l,0}, \mathsf{info}_{l,0})_{l\in[L_0]}) \text{ and } \mathsf{U}(\mathsf{pk}, L_1, (\mathsf{m}_{l,1}, \mathsf{info}_{l,1})_{l\in[L_1]}),$$

   respectively. Run $\mathcal{A}$ *on input* St *with arbitrary interleaved one-time access to each of these oracles, i.e.,* $St' \leftarrow \mathcal{A}^{O_0,O_1}(St)$*.*
4. *Let*

$$\sigma_{1,0}, \ldots, \sigma_{l_{*,0}-1,0}, \sigma_{l_{*,b},b}, \sigma_{l_{*,0}+1,0}, \ldots, \sigma_{L_0} \text{ and}$$
$$\sigma_{1,1}, \ldots, \sigma_{l_{*,1}-1,1}, \sigma_{l_{*,1-b},1-b}, \sigma_{l_{*,1}+1,1}, \ldots, \sigma_{L_1}$$

   be the local outputs of O_0, O_1*, respectively. If* $\sigma_{i,0} = \bot$ *or* $\sigma_{i',1} = \bot$ *for some* $i \in [L_0]$ *or some* $i' \in [L_1]$*, then run* $b' \leftarrow \mathcal{A}(St', \bot)$*. Else, obtain a bit* b' *from* $\mathcal{A}$ *on input* $(\sigma_{l,0})_{l\in[L_0]}, (\sigma_{l,1})_{l\in[L_1]}$*. That is, run* $b' \leftarrow \mathcal{A}\left(St', (\sigma_{l,0})_{l\in[L_0]}, (\sigma_{l,1})_{l\in[L_1]}\right)$*.*
5. *Output* b'*.*

We say that BPBS *satisfies malicious signer batch partial blindness, if for every PPT algorithm* $\mathcal{A}$ *the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{bblind}}_{\mathcal{A},\mathsf{BPBS}}(\lambda) := \left|\Pr\left[\mathbf{BBLIND}^{\mathcal{A}}_{0,\mathsf{BPBS}}(\lambda) \Rightarrow 1\right] - \Pr\left[\mathbf{BBLIND}^{\mathcal{A}}_{1,\mathsf{BPBS}}(\lambda) \Rightarrow 1\right]\right|.$$

Observe that batched partially blind signatures imply partially blind signatures by fixing the batch size $L = 1$. Further, the partial blindness can be lifted to standard blindness by fixing a default public information string.

3.4 PI-Cut-Choo Blind Signatures

Here, we construct a concrete blind signature scheme $\mathsf{BS}_{\mathsf{CDH}}$ based on the CDH assumption in the type-1 pairing setting. The scheme and its analysis can easily be adopted to the type-3 setting, which is more relevant in practice. However, as we present a much more practical scheme from similar assumptions in Section 3.7, we use the type-1 setting in this section for readability.

3.4.1 Construction

Let PGGen be a bilinear group generation algorithm that outputs a cyclic group $\mathbb{G}$ of prime order p with generator g, and a pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. We assume that these system parameters are known to all algorithms. Formally, they should be part of the public key, but as they are standardized and their correctness can be verified efficiently in practice, and for readability, we omit them from the public key. Our scheme makes use of a randomness homomorphic commitment scheme CMT with randomness space $\mathcal{R}_{\mathsf{ck}}$ and a puncturable pseudorandom function PRF. As we have seen, we can instantiate PRF using random oracles and CMT tightly based on the DLOG assumption. We also need random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}'\colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_x\colon \{0,1\}^* \to \mathbb{Z}_p \times \mathcal{R}_{\mathsf{ck}} \times \{0,1\}^{\lambda_{\mathsf{PRF}}}$, where λ_{PRF} is a security parameter used for PRF. Further, our scheme makes use of a parameter $K \in \mathbb{N}$, which defines how many

Alg $\mathsf{Check}(\mathsf{pk}, N, \mu_0, \mathsf{com}_\mathsf{r}, \mathsf{com}_c, \mathsf{seed}_\mathbf{J}, k_\mathbf{J}, \{c_{i,\mathbf{J}_i}\}_i, \{\eta_i\}_i)$

$\mathbf{J} = (\mathsf{H}'(\mathsf{seed}_\mathbf{J}, 1), \ldots, \mathsf{H}'(\mathsf{seed}_\mathbf{J}, K)) \in [N]^K$
for $i \in [K]$:
  for $j \in [N] \setminus \{\mathbf{J}_i\}$:
    $\mathsf{prer}_{i,j} := \mathsf{PRF.Eval}(k_\mathbf{J}, (i,j)),\ \mathsf{r}_{i,j} := \mathsf{H}_x(\mathsf{prer}_{i,j})$
    parse $(\alpha_{i,j}, \varphi_{i,j}, \gamma_{i,j}) := \mathsf{r}_{i,j},\ (\alpha_{i,j}, \varphi_{i,j}, \gamma_{i,j}) \in \mathbb{Z}_p \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^\lambda$
    $\mu_{i,j} := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi_{i,j})$
    $c_{i,j} := \mathsf{H}(\mathsf{pk}_i, \mu_{i,j}) \cdot g^{\alpha_{i,j}}$
  $\mathsf{com}_{\mathsf{r},i} := \mathsf{H}_\mathsf{r}(\mathsf{H}_\mathsf{r}(\mathsf{r}_{i,1}), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,\mathbf{J}_i-1}), \eta_i, \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,\mathbf{J}_i+1}), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,N}))$
if $\mathsf{com}_\mathsf{r} \neq \mathsf{H}_\mathsf{r}(\mathsf{com}_{\mathsf{r},1}, \ldots, \mathsf{com}_{\mathsf{r},K})$: **return** 0
if $\mathsf{com}_c \neq \mathsf{H}_c(c_{1,1}, \ldots, c_{K,N})$: **return** 0
return 1

Figure 3.1: Algorithm Check used in the issuing protocol of blind signature scheme $\mathsf{BS}_{\mathsf{CDH}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}, \mathsf{H}'\colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_x\colon \{0,1\}^* \to \mathbb{Z}_p \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$ are random oracles.

instances of the underlying boosting transform are executed in parallel, and a function $f\colon \mathbb{N} \to \mathbb{N}$ which determines how fast the cut-and-choose parameter N grows. In Section 3.4.2, we will discuss how to set these parameters efficiently.

Key Generation. To generate keys, algorithm $\mathsf{BS}_{\mathsf{CDH}}.\mathsf{Gen}(1^\lambda)$ does the following:

1. For each instance $i \in [K]$, sample $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$ and set $\mathsf{pk}_i := g^{\mathsf{sk}_i}$.
2. Sample a commitment key $\mathsf{ck} \leftarrow \mathsf{CMT.Gen}(1^\lambda)$.
3. Return public key $\mathsf{pk} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{ck})$ and secret key $\mathsf{sk} := (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)$.

Signature Issuing. The algorithms S, U and their interaction are formally given in Figures 3.1 and 3.2. Here, the Signer is stateful. Precisely, algorithm S keeps a state ctr, which is inititalized as $\mathsf{ctr} := 1$ and incremented in every interaction.

Verification. The resulting signature $\sigma = (\bar{\sigma}, \varphi_1, \ldots, \varphi_K)$ for a message m is verified by algorithm $\mathsf{BS}_{\mathsf{CDH}}.\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$ as follows:

1. For each instance $i \in [K]$, compute the commitment $\mu_i := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_i)$.
2. Return 1 if and only if $e(\bar{\sigma}, g) = \prod_{i=1}^K e(\mathsf{H}(\mathsf{pk}_i, \mu_i), \mathsf{pk}_i)$.

Analysis. Completeness of the scheme follows by inspection. We show blindness and one-more unforgeability. For one-more unforgeability, we show $q_{\max}$-OMUF, where $q_{\max}$ is a parameter that can be set freely, e.g., $q_{\max} = 2^{30}$, and influences the function f. We note that making f grow quadratically, one could show full OMUF using a similar proof. Before we show blindness, we first state and prove a lemma that will be useful for our blindness proof.

Lemma 3.3. *For any algorithm $\mathcal{A}$, parameters $\mathsf{par} := (\mathbb{G}, g, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, and bit $b \in \{0,1\}$, we consider the following game $\boldsymbol{G}_b$:*

1. *Let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$[5]. Run $((\mathsf{pk}_i, \mathsf{m}_{i,0}, \mathsf{m}_{i,1})_{i\in[K]}, St) \leftarrow \mathcal{A}^\mathsf{H}(\mathsf{par})$.*
2. *Let $O_{b'}$ for $b' \in \{0,1\}$ be an interactive oracle. Upon termination, it locally outputs $\sigma_{b\oplus b'}$ to the game. The oracle is defined as follows:*

[5] We do not need to model H as a random oracle here.

$\underline{\mathsf{S}(\mathsf{sk})}$; state ctr		$\underline{\mathsf{U}(\mathsf{pk}, \mathsf{m})}$
$\mathsf{ctr} := \mathsf{ctr} + 1, N := f(\mathsf{ctr})$		$k \leftarrow \mathsf{PRF.Gen}(1^{\lambda_{\mathsf{PRF}}}, 1^{\log(KN)})$
$\mathsf{seed}_{\mathbf{J}}, \mathsf{salt} \xleftarrow{\$} \{0,1\}^{\lambda}$		$\varphi_0 \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}, \mu_0 := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_0)$
$\mathsf{com}_{\mathbf{J}} := \mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, \mathsf{salt})$	$\xrightarrow{N, \mathsf{com}_{\mathbf{J}}}$	**for** $(i,j) \in [K] \times [N]$:
		$\mathsf{prer}_{i,j} := \mathsf{PRF.Eval}(k, (i,j))$
		$\mathsf{r}_{i,j} := \mathsf{H}_x(\mathsf{prer}_{i,j})$
		parse $(\alpha_{i,j}, \varphi_{i,j}, \gamma_{i,j}) := \mathsf{r}_{i,j}$
		$\mu_{i,j} := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi_{i,j})$
		$c_{i,j} := \mathsf{H}(\mathsf{pk}_i, \mu_{i,j}) \cdot g^{\alpha_{i,j}}$
		for $i \in [K]$:
		$\mathsf{com}_{\mathsf{r},i} := \mathsf{H}_{\mathsf{r}}(\mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,1}), \ldots, \mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,N}))$
		$\mathsf{com}_{\mathsf{r}} := \mathsf{H}_{\mathsf{r}}(\mathsf{com}_{\mathsf{r},1}, \ldots, \mathsf{com}_{\mathsf{r},K})$
	$\xleftarrow{\mu_0, \mathsf{com}_{\mathsf{r}}, \mathsf{com}_c}$	$\mathsf{com}_c := \mathsf{H}_c(c_{1,1}, \ldots, c_{K,N})$
	$\xrightarrow{\mathsf{seed}_{\mathbf{J}}, \mathsf{salt}}$	**if** $\mathsf{com}_{\mathbf{J}} \neq \mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, \mathsf{salt})$: **abort**
for $i \in [K]$:		**for** $i \in [K]$: $\mathbf{J}_i := \mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, i)$
$\mathbf{J}_i := \mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, i)$		$\mathbf{J} = (\mathbf{J}_1, \ldots, \mathbf{J}_K)$
$\mathbf{J} = (\mathbf{J}_1, \ldots, \mathbf{J}_K)$		$\mathcal{J} := \{(i, \mathbf{J}_i) \mid i \in [K]\}$
		$k_{\mathbf{J}} \leftarrow \mathsf{PRF.Puncture}(k, \mathcal{J})$
if $\mathsf{Check} = 0$: **abort**	$\xleftarrow{k_{\mathbf{J}}, \{c_{i,\mathbf{J}_i}, \eta_i\}_i}$	**for** $i \in [K]$: $\eta_i := \mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,\mathbf{J}_i})$
for $i \in [K]$: $s_{i,\mathbf{J}_i} := c_{i,\mathbf{J}_i}^{\mathsf{sk}_i}$		
$\bar{s} := \prod_{i=1}^{K} s_{i,\mathbf{J}_i}$	$\xrightarrow{\bar{s}}$	$\bar{\sigma} := \bar{s} \cdot \prod_{i=1}^{K} \mathsf{pk}_i^{-\alpha_{i,\mathbf{J}_i}}$
		if $\prod_{i=1}^{K} e\left(\mathsf{H}(\mathsf{pk}_i, \mu_{i,\mathbf{J}_i}), \mathsf{pk}_i\right) \neq e\left(\bar{\sigma}, g\right)$: **abort**
		for $i \in [K]$: $\varphi_i := \varphi_0 + \varphi_{i,\mathbf{J}_i}$
		return $\sigma := (\bar{\sigma}, \varphi_1, \ldots, \varphi_K)$

Figure 3.2: The signature issuing protocol of the blind signature scheme $\mathsf{BS}_{\mathsf{CDH}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}'\colon \{0,1\}^* \to \{0,1\}^{\lambda}$ and $\mathsf{H}_{\mathsf{r}}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^{\lambda}, \mathsf{H}_x\colon \{0,1\}^* \to \mathbb{Z}_p \times \mathcal{R}_{\mathsf{ck}} \times \{0,1\}^{\lambda_{\mathsf{PRF}}}$ are random oracles. The algorithm Check is defined in Figure 3.1. The state ctr of S is incremented atomically.

(a) *Upon a query from* $\mathcal{A}$*, sample* $\alpha_i \xleftarrow{\$} \mathbb{Z}_p$ *and set* $c_i := \mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,b\oplus b'}) \cdot g^{\alpha_i}$ *for all* $i \in [K]$*. Send* $c_1, \ldots, c_K$ *to* $\mathcal{A}$*.*

(b) *Receive* $\bar{s}$ *from* $\mathcal{A}$ *and set* $\bar{\sigma} := \bar{s} \cdot \prod_{i=1}^{K} \mathsf{pk}_i^{-\alpha_i}$*. If* $e(\bar{\sigma}, g) \neq \prod_{i\in[K]} e(\mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,b\oplus b'}), \mathsf{pk}_i)$*, define the local output of this oracle to be* $\sigma_{b\oplus b'} := \bot$*. Otherwise, define the local output of this oracle to be* $\sigma_{b\oplus b'} := \bar{\sigma}$*.*

3. *Run* $\mathcal{A}$ *on input* St *with arbitrary interleaved one-time access to each of these oracles, i.e.,* $St' \leftarrow \mathcal{A}^{O_0, O_1, \mathsf{H}}(St)$*.*

4. *If* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$*, run* $b^* \leftarrow \mathcal{A}(St', \bot, \bot)$*. Else, run* $b^* \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$*. Output* b^**.*

Then, for each algorithm $\mathcal{A}$*, we have* $\Pr[\boldsymbol{G}_0 \Rightarrow 1] = \Pr[\boldsymbol{G}_1 \Rightarrow 1]$.

Proof. We show the claim via a statistical argument. To this end, recall that the exponentiation map $\mathbb{Z}_p \to \mathbb{G}, x \mapsto g^x$ is a bijection. Thus, for each public key pk_i output by $\mathcal{A}$, we can write $\mathsf{pk}_i = g^{\mathsf{sk}_i}$. For now, we denote the discrete logarithm of an element $h \in \mathbb{G}$ with respect to g by $\mathsf{dlog}(h)$. With this notation, we have $\mathsf{sk}_i = \mathsf{dlog}(\mathsf{pk}_i)$ for all $i \in [K]$. After the adversary outputs $(\mathsf{pk}_i, \mathsf{m}_{i,0}, \mathsf{m}_{i,1})_{i\in[K]}$, we consider the rest of the experiment in two phases. First, we consider the view of $\mathcal{A}$ before it receives σ_0 and σ_1. Here, it is clear that $\mathcal{A}$'s view in $\mathbf{G}_b$ is the same for both $b = 0$ and $b = 1$. Indeed, in both games, $\mathcal{A}$ obtains from both oracles K independent and uniform group elements c_i, as the values α_i act as a one-time pad, hiding $\mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,b\oplus b'})$ and thus b. Next, we consider the view of $\mathcal{A}$ after it receives σ_0 and σ_1. If $\sigma_0 = \bot$ or $\sigma_1 = \bot$, then $\mathcal{A}$ obtains no new information about b. On the other hand, if $\sigma_0 \neq \bot$ and $\sigma_1 \neq \bot$ we know that, by definition of game $\mathbf{G}_b$, we have

$$
\begin{aligned}
& e(\sigma_j, g) = \prod_{i\in[K]} e(\mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,j}), \mathsf{pk}_i) \\
\iff \quad & e(g,g)^{\mathsf{dlog}_g(\sigma_j)} = e(g,g)^{\sum_{i\in[K]} \mathsf{dlog}(\mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,j}))\mathsf{sk}_i} \\
\iff \quad & \sigma_j = \prod_{i\in[K]} \mathsf{H}(\mathsf{pk}_i, \mathsf{m}_{i,j})^{\mathsf{sk}_i}.
\end{aligned}
$$

for each $j \in \{0,1\}$, where the last equivalence follows from the non-degeneracy of the pairing. Thus, for each $j \in \{0,1\}$, the element σ_j is completely determined by $(\mathsf{pk}_i, \mathsf{m}_{i,j})_{i\in[K]}$. This implies that after learning σ_0 and σ_1, $\mathcal{A}$ obtains no additional information about bit b. Therefore, the claim follows. □

Theorem 3.1. *Let* PRF *be a puncturable pseudorandom function and* CMT *be a randomness homomorphic commitment scheme. Let* $\mathsf{H}' \colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\mathsf{r} \colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_x \colon \{0,1\}^* \to \mathbb{Z}_p \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$ *be random oracles. Then* BS_CDH *satisfies malicious signer blindness.*

In particular, for any algorithm $\mathcal{A}$ *that uses* N^L *and* N^R *as the counters in its interactions with the User and queries* $\mathsf{H}', \mathsf{H}_\mathsf{r}, \mathsf{H}_x$ *at most* $Q_{\mathsf{H}'}, Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_x}$ *times, respectively, there is an algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$
\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}_\mathsf{CDH}}(\lambda) \leq 4 \cdot \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda) + \frac{Q^2_{\mathsf{H}'}}{2^{\lambda-1}} + \frac{Q_{\mathsf{H}'}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_x}}{2^{\lambda_\mathsf{PRF}-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda_\mathsf{PRF}-2}},
$$

where $\mathcal{B}$ *punctures at* K *points with input length* $d = \max\{\log(N^L), \log(N^R)\}$.

Proof. Let $\mathcal{A}$ be an adversary against malicious signer blindness of $\mathsf{BS} := \mathsf{BS}_\mathsf{CDH}$. We show the claimed uppor bound on its advantage via a sequence of games, where all random oracles are simulated honestly via lazy sampling unless otherwise specified.

Game $\mathbf{G}_{0,b}$: Game $\mathbf{G}_{0,b}$ is defined as the real blindness game $\mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}$. Let us recall this game. First, $\mathcal{A}$ outputs a public key pk and messages $\mathsf{m}_0, \mathsf{m}_1$. Then, the game provides two interactive oracles $\mathrm{O}_0, \mathrm{O}_1$ to $\mathcal{A}$, which simulate the user algorithm $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b), \mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$, respectively. Throughout the proof,

we will reference to the variables used in these executions using superscripts L and R, respectively. For example, $\mathsf{com}_{\mathbf{J}}^{L}$ refers to the commitment on the seed of the cut-and-choose index sent by $\mathcal{A}$ as part of the first message in the interaction with oracle O_0. If we omit the superscript, our description applies to both oracles. According to this, N^L and N^R denote the cut-and-choose parameters sent by $\mathcal{A}$ in the first message of the interaction with oracles $\mathrm{O}_0, \mathrm{O}_1$, respectively. It follows that

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \left|\Pr\left[\mathbf{G}_{0,0} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{0,1} \Rightarrow 1\right]\right|.$$

Game $\mathbf{G}_{1,b}$: Game $\mathbf{G}_{1,b}$ is exactly as game $\mathbf{G}_{0,b}$, but whenever there are queries $\mathsf{H}'(x) = \mathsf{H}'(x')$ for $x \neq x'$, the game aborts. Clearly, the probability of such a collision is at most $Q^2_{\mathsf{H}'}/2^\lambda$, which leads to

$$\left|\Pr\left[\mathbf{G}_{0,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right]\right| \leq \frac{Q^2_{\mathsf{H}'}}{2^\lambda}.$$

Game $\mathbf{G}_{2,b}$: Game $\mathbf{G}_{2,b}$ is exactly as game $\mathbf{G}_{1,b}$, but we introduce another abort. Namely, the game aborts if the adversary sends $N, \mathsf{com}_{\mathbf{J}}$ as its first message, but at that point the game can not find a query $\mathsf{H}'(\hat{\mathsf{seed}}_{\mathbf{J}}, \hat{\mathsf{salt}}) = \mathsf{com}_{\mathbf{J}}$ and later the user algorithm does not abort, i.e., $\mathcal{A}$ is able to successfully open $\mathsf{com}_{\mathbf{J}}$ by sending $\mathsf{seed}_{\mathbf{J}}, \mathsf{salt}$. Note that there is at most one query that the game can find, as we ruled out collisions for oracle H' in the previous change. Clearly, the probability that the adversary can successfully open a commitment for which the game can not find a query is at most $Q_{\mathsf{H}'}/2^\lambda$. A union bound over oracles O_0 and O_1 shows that

$$\left|\Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{2,b} \Rightarrow 1\right]\right| \leq \frac{Q_{\mathsf{H}'}}{2^{\lambda-1}}.$$

Note that if the user oracle aborts, then the adversary gets $(\bot, \bot)$ in the end of the game and learns nothing about the bit b as CMT is perfectly hiding and no information about the randomness φ_0 is ever revealed to $\mathcal{A}$. Thus, from now on, we can focus on the case where the user oracle does not abort. By the change we introduced here, we can thus assume that the game is able to extract $\hat{\mathsf{seed}}_{\mathbf{J}}$ from $\mathsf{com}_{\mathbf{J}}$ and that later $\hat{\mathsf{seed}}_{\mathbf{J}} = \mathsf{seed}_{\mathbf{J}}$. For the extracted seed $\hat{\mathsf{seed}}_{\mathbf{J}}$, we also define the cut-and-choose vector $\hat{\mathbf{J}}$ and the set $\hat{\mathcal{J}}$ as

$$\forall i \in [K] : \hat{\mathbf{J}}_i := \mathsf{H}'(\hat{\mathsf{seed}}_{\mathbf{J}}, i), \;\; \hat{\mathbf{J}} = (\hat{\mathbf{J}}_1, \ldots, \hat{\mathbf{J}}_K), \;\; \hat{\mathcal{J}} := \{(i, \hat{\mathbf{J}}_i) \mid i \in [K]\}.$$

As $\hat{\mathsf{seed}}_{\mathbf{J}} = \mathsf{seed}_{\mathbf{J}}$, we also have $\hat{\mathbf{J}} = \mathbf{J}$ and $\hat{\mathcal{J}} = \mathcal{J}$.

Game $\mathbf{G}_{3,b}$: Game $\mathbf{G}_{3,b}$ is defined exactly as $\mathbf{G}_{2,b}$, except that we change the way the randomness seeds $\mathsf{prer}_{i,j}$ are generated. We recall that in previous games, these values were generated as in the real scheme, i.e.,

$$\mathsf{prer}_{i,j} := \mathsf{PRF.Eval}(k, (i,j)) \text{ for all } (i,j) \in [K] \times [N].$$

Instead, we now generate these values using a punctured key $k_{\hat{\mathbf{J}}}$ for $(i,j) \in [K] \times [N] \setminus \hat{\mathcal{J}}$, and as before for $(i,j) \in \hat{\mathcal{J}}$. Concretely, at the beginning of the interaction, the game samples $k \leftarrow \mathsf{PRF.Gen}(1^{\lambda_{\mathsf{PRF}}}, 1^{\log(KN)})$ as before, extracts $\hat{\mathsf{seed}}_{\mathbf{J}}$ and computes $\hat{\mathcal{J}}$ as described in $\mathbf{G}_{2,b}$, and additionally generates $k_{\hat{\mathbf{J}}} \leftarrow \mathsf{PRF.Puncture}(k, \hat{\mathcal{J}})$. Then it sets

$$\mathsf{prer}_{i,\hat{\mathbf{J}}_i} := \mathsf{PRF.Eval}(k, (i, \hat{\mathbf{J}}_i)) \text{ for all } i \in [K]$$

and

$$\mathsf{prer}_{i,j} := \mathsf{PRF.Eval}(k_{\hat{\mathbf{J}}}, (i,j)) \text{ for all } (i,j) \in [K] \times [N] \setminus \hat{\mathcal{J}}.$$

By the completeness of PRF this is only a syntactical change, and hence

$$\Pr\left[\mathbf{G}_{3,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{2,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{4,b}$: In game $\mathbf{G}_{4,b}$, we change the way we generate the randomness seeds $\mathsf{prer}^{L}_{i,\hat{\mathbf{J}}^L_i}$ for $i \in [K]$. Concretely, we sample them at random from $\{0,1\}^{\lambda}_{\mathsf{PRF}}$. We can bound the distinguishing advantage between games $\mathbf{G}_{3,b}$ and $\mathbf{G}_{4,b}$ using a reduction $\mathcal{B}$ from the security of PRF. The reduction $\mathcal{B}$ is as follows:

- Run $\mathcal{A}$ to get a public key and messages, i.e., $(\mathsf{pk}, \mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}(1^\lambda)$.
- Run $\mathcal{A}$ on input St with access to random oracles and interactive oracles $\mathrm{O}_0, \mathrm{O}_1$, i.e., $St' \leftarrow \mathcal{A}^{\mathrm{O}_0,\mathrm{O}_1}(St)$. The oracle O_1 is provided as in game $\mathbf{G}_{3,b}$ and oracle O_0 is provided as follows:
  - When $\mathcal{A}$ sends $N^L, \mathsf{com}_{\mathbf{J}}^L$, extract $\hat{\mathbf{J}}^L, \hat{\mathcal{J}}^L$ from com_J^L as game $\mathbf{G}_{3,b}$ does and output $\hat{\mathcal{J}}^L$ to the PRF challenger. Obtain the punctured key $k_{\hat{\mathbf{J}}^L}$ and values $\{\mathsf{prer}^L_{i,\hat{\mathbf{J}}_i^L}\}_{i\in[K]}$.
  - Use $k_{\hat{\mathbf{J}}^L}$ to sample $\mathsf{prer}^L_{i,j}$ for $(i,j) \in [K] \times [N^L] \setminus \hat{\mathcal{J}}^L$ as in $\mathbf{G}_{3,b}$. Continue the oracle simulation as in $\mathbf{G}_{3,b}$. According to this, if the simulation does not abort, send the key $k_{\hat{\mathbf{J}}^L}$ in the fourth message of the interaction.
- Let σ_b, σ_{1-b} be the local outputs of $\mathrm{O}_0, \mathrm{O}_1$, respectively. If $\sigma_0 = \bot$ or $\sigma_1 = \bot$, then run $b' \leftarrow \mathcal{A}(St', \bot, \bot)$. Else, run $b' \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$ and output b'.

Note that if the values $\mathsf{prer}^L_{i,\hat{\mathbf{J}}_i^L}$ are random, then $\mathcal{B}$ perfectly simulates $\mathbf{G}_{4,b}$, whereas if they are the outputs of the pseudorandom function, $\mathcal{B}$ perfectly simulates $\mathbf{G}_{3,b}$. By the security of PRF with input length $\log(KN^L)$ we obtain

$$|\Pr[\mathbf{G}_{3,b} \Rightarrow 1] - \Pr[\mathbf{G}_{4,b} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda).$$

Game $\mathbf{G}_{5,b}$: In game $\mathbf{G}_{5,b}$, we change the way we generate the randomness seeds $\mathsf{prer}^R_{i,\hat{\mathbf{J}}_i^R}$ for $i \in [K]$. Concretely, we sample them at random from $\{0,1\}^\lambda_{\mathsf{PRF}}$. Anologously to the previous change, a reduction from the security of PRF shows that

$$|\Pr[\mathbf{G}_{4,b} \Rightarrow 1] - \Pr[\mathbf{G}_{5,b} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda).$$

Game $\mathbf{G}_{6,b}$: In game $\mathbf{G}_{6,b}$, we change the way we compute the values $\mathsf{r}_{i,\hat{\mathbf{J}}_i}$ for $i \in [K]$. Note that in $\mathbf{G}_{5,b}$ these were computed as $\mathsf{r}_{i,\hat{\mathbf{J}}_i} := \mathsf{H}_x(\mathsf{prer}_{i,\hat{\mathbf{J}}_i})$. Now, we sample them randomly as

$$\mathsf{r}_{i,\hat{\mathbf{J}}_i} = (\alpha_{i,\hat{\mathbf{J}}_i}, \varphi_{i,\hat{\mathbf{J}}_i}, \gamma_{i,\hat{\mathbf{J}}_i}) \xleftarrow{\$} \mathbb{Z}_p \times \mathcal{R}_{\mathsf{ck}} \times \{0,1\}^{\lambda_{\mathsf{PRF}}}.$$

Note that $\mathcal{A}$ can only distinguish between games $\mathbf{G}_{5,b}$ and $\mathbf{G}_{6,b}$ if it queries $\mathsf{H}_x(\mathsf{prer}_{i,\hat{\mathbf{J}}_i})$ for some $i \in [K]$. However, $\mathcal{A}$ obtains no information about $\mathsf{prer}_{i,\hat{\mathbf{J}}_i}$, which is sampled uniformly at random. By a union bound over all hash queries, $i \in [K]$ and $\{L, R\}$ we obtain

$$|\Pr[\mathbf{G}_{5,b} \Rightarrow 1] - \Pr[\mathbf{G}_{6,b} \Rightarrow 1]| \leq \frac{2KQ_{\mathsf{H}_x}}{2^{\lambda_{\mathsf{PRF}}}}.$$

Game $\mathbf{G}_{7,b}$: Game $\mathbf{G}_{7,b}$ is as $\mathbf{G}_{6,b}$, except that it computes the values $\mathsf{com}_{\mathsf{r},i}, i \in [K]$ in a different way. Concretely, for all $i \in [K]$, it samples $\eta_i \xleftarrow{\$} \{0,1\}^\lambda$ and computes the $\mathsf{com}_{\mathsf{r},i}$ as

$$\mathsf{com}_{\mathsf{r},i} := \mathsf{H}_{\mathsf{r}}(\mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,1}), \ldots, \mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,\hat{\mathbf{J}}_i-1}), \eta_i, \mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,\hat{\mathbf{J}}_i+1}), \ldots, \mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,N})).$$

Later it returns the η_i as part of its second message. Note that $\mathcal{A}$ can only see the difference between $\mathbf{G}_{6,b}$ and $\mathbf{G}_{7,b}$ if it queries $\mathsf{H}_{\mathsf{r}}(\mathsf{r}^X_{i,\hat{\mathbf{J}}_i^X})$ for an $i \in [K]$ and $X \in \{L, R\}$. It is clear that $\mathcal{A}$ obtains no information about $\gamma_{i,\hat{\mathbf{J}}_i}$ and $\gamma_{i,\hat{\mathbf{J}}_i}$ is sampled uniformly at random. Thus, a union bound over all $Q_{\mathsf{H}_{\mathsf{r}}}$ random oracle queries, $i \in [K]$, and $X \in \{L, R\}$ yields

$$|\Pr[\mathbf{G}_{6,b} \Rightarrow 1] - \Pr[\mathbf{G}_{7,b} \Rightarrow 1]| \leq \frac{2KQ_{\mathsf{H}_{\mathsf{r}}}}{2^{\lambda_{\mathsf{PRF}}}}.$$

Game $\mathbf{G}_{8,b}$: In game $\mathbf{G}_{8,b}$ we change the way the commitments $\mu_{i,\hat{\mathbf{J}}_i}, i \in [K]$ are generated. Recall that before, these were generated as

$$\mu_{i,\hat{\mathbf{J}}_i} := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi_{i,\hat{\mathbf{J}}_i}) = \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_0 + \varphi_{i,\hat{\mathbf{J}}_i}).$$

Note that if the game does not stop, then especially $\hat{\mathbf{J}} = \mathbf{J}$ and $\varphi_i = \varphi_0 + \varphi_{i,\hat{\mathbf{J}}_i}$. In game $\mathbf{G}_{8,b}$, we sample $\varphi_i \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}$ and set $\mu_{i,\hat{\mathbf{J}}_i} := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_i)$ for all $i \in [K]$. We claim that the view of $\mathcal{A}$ is unchanged. This is because, due to the previous changes, $\mathcal{A}$ gets no information about $\varphi_{i,\hat{\mathbf{J}}_i}$. Thus, we have to consider the distribution of the values $\varphi_i = \varphi_0 + \varphi_{i,\hat{\mathbf{J}}_i}$ conditioned on $k_{\hat{\mathbf{J}}}, (\varphi_0 + \varphi_{i,j})_{j \neq \hat{\mathbf{J}}_i}$ and φ_0. This distribution is uniformly random as $\varphi_{i,\hat{\mathbf{J}}_i}$ is uniformly random. Hence we have

$$\Pr[\mathbf{G}_{8,b} \Rightarrow 1] = \Pr[\mathbf{G}_{7,b} \Rightarrow 1].$$

Game $\mathbf{G}_{9,b}$: In game $\mathbf{G}_{9,b}$, we change the way μ_0 is generated, using that CMT is perfectly hiding. Concretely, we sample a random message $\bar{\mathsf{m}}^L$ (resp. $\bar{\mathsf{m}}^R$) and set $\mu_0 := \mathsf{Com}(\mathsf{ck}, \bar{\mathsf{m}}; \varphi_0)$. Note that in $\mathbf{G}_{9,b}$ the value φ_0 is only needed to compute μ_0. Especially, it is not needed to compute the values φ_i which are part of the final signatures due to the previous changes. It follows from the security of CMT that $\mathsf{Com}(\mathsf{ck}, \bar{\mathsf{m}}; \varphi_0)$ and $\mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_0)$ are identically distributed given ck. Therefore, the view of $\mathcal{A}$ is not changed and we get

$$\Pr[\mathbf{G}_{9,b} \Rightarrow 1] = \Pr[\mathbf{G}_{8,b} \Rightarrow 1].$$

Let us take a closer look at game $\mathbf{G}_{9,b}$. Here, for each instance $i \in [K]$, the only part that depends on message m (and hence bit b) is the $\mathbf{J}_i^{th}$ session. All other sessions only depend on μ_0, which does not depend on m anymore. Now, our final claim is that we can bound the difference between $\mathbf{G}_{9,0}$ and $\mathbf{G}_{9,1}$ using the perfect blindness under maliciously generated keys of the well-known BLS blind signatures scheme [Bol03]. Concretely, we can now apply a straight-forward reduction from the game in Lemma 3.3 and obtain

$$\Pr[\mathbf{G}_{9,0} \Rightarrow 1] = \Pr[\mathbf{G}_{9,1} \Rightarrow 1].$$

In summary, we showed that $\mathbf{G}_{0,0}$ is close to $\mathbf{G}_{9,0}$, $\mathbf{G}_{9,0}$ is close to $\mathbf{G}_{9,1}$, and $\mathbf{G}_{9,1}$ is close to $\mathbf{G}_{0,1}$. Thus, $\mathbf{G}_{0,0}$ is close to $\mathbf{G}_{0,1}$, which is what we had to show. □

Theorem 3.2. *Let* CMT *be a randomness homomorphic commitment scheme and* PRF *be a puncturable pseudorandom function. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}'\colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda$ *be random oracles. Also, assume that there is a* $\vartheta > 0$ *and* f *is such that*

$$f(\mathsf{ctr}) = \lceil 3\vartheta \ln(q_{\max} + 1) \cdot \mathsf{ctr} \rceil .$$

Then, $\mathsf{BS}_{\mathsf{CDH}}$ *satisfies* $q_{\max}$*-one-more unforgeability, under the* CDH *assumption relative to* PGGen.

Specifically, let $\delta > 0$ *such that* $(1-\delta)\vartheta > 1$*. Then, for any PPT algorithm* $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_c}, Q_{\mathsf{H}'}, Q_{\mathsf{H}}$ *queries to oracles* $\mathsf{H}_\mathsf{r}, \mathsf{H}_c, \mathsf{H}', \mathsf{H}$*, respectively, and starts at most* $q \leq q_{\max}$ *interactions with his signer oracle, there are PPT algorithms* $\mathcal{B}$ *and* $\mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{q_{\max}\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}_{\mathsf{CDH}}}(\lambda) - e^{-\delta K} \leq \mathsf{Adv}^{\mathsf{bind}}_{\mathcal{B}',\mathsf{CMT}}(\lambda) + \frac{K}{p} + 4qK \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda) + \mathsf{stat}$$

where

$$\mathsf{stat} = \frac{Q_{\mathsf{H}_\mathsf{r}}^2}{2^\lambda} + \frac{Q_{\mathsf{H}_c}^2}{2^\lambda} + \frac{qQ_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{qKQ_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{qQ_{\mathsf{H}_c}}{2^\lambda} + \frac{qQ_{\mathsf{H}'}}{2^{\lambda-1}}.$$

Proof. Set $\mathsf{BS} := \mathsf{BS}_{\mathsf{CDH}}$. Let $\mathcal{A}$ be an adversary against the OMUF security of BS. We prove the statement via a sequence of games.

Game $\mathbf{G}_0$: We start with game $\mathbf{G}_0 := q_{\max}\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}$, which is the one-more unforgeability game. We briefly recall this game. A key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is sampled, $\mathcal{A}$ is run with concurrent access to an interactive oracle O simulating $\mathsf{S}(\mathsf{sk})$. Assume that $\mathcal{A}$ completes ℓ interactions with O. Further, $\mathcal{A}$ gets access to random oracles $\mathsf{H}, \mathsf{H}', \mathsf{H}_\mathsf{r}$ and H_c, which are provided by the game in the standard lazy

manner. When $\mathcal{A}$ finishes its execution, it outputs tuples $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$ and wins, if all m_i are distinct, $k > \ell$ and all signatures σ_i verify with respect to pk and m_i.

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, we add an additional abort. The game aborts if in the end $\mathcal{A}$'s output contains two pairs $(\mathsf{m}^{(0)}, \sigma^{(0)}), (\mathsf{m}^{(1)}, \sigma^{(1)})$ such that $\mathsf{m}^{(0)} \neq \mathsf{m}^{(1)}$ but there exists $i^{(0)}, i^{(1)} \in [K]$ such that

$$\mathsf{Com}(\mathsf{ck}, \mathsf{m}^{(0)}; \varphi^{(0)}_{i^{(0)}}) = \mathsf{Com}(\mathsf{ck}, \mathsf{m}^{(1)}; \varphi^{(1)}_{i^{(1)}}).$$

As CMT is computationally binding, a straight-forward reduction $\mathcal{B}'$ shows that

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{bind}}_{\mathcal{B}', \mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_2$: This game is as $\mathbf{G}_1$, but we rule out collisions for oracles $\mathsf{H}_t, t \in \{\mathsf{r}, c\}$. To be more precise, we change the simulation of oracles $\mathsf{H}_t, t \in \{\mathsf{r}, c\}$ in the following way. If $\mathcal{A}$ queries $\mathsf{H}_t(x)$ and this value is not yet defined, the game samples an image $y \xleftarrow{\$} \{0,1\}^\lambda$. However, if there exists an $x' \neq x$ with $\mathsf{H}_t(x') = y$, the game returns $\perp$. Otherwise it behaves as before. Note that $\mathcal{A}$ can only distinguish between $\mathbf{G}_1$ and $\mathbf{G}_2$ if such a collision happens, i.e., H_t returns $\perp$. We can apply a union bound over all $Q^2_{\mathsf{H}_t}$ pairs of random oracle queries and obtain

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_\mathsf{c}}}{2^\lambda}.$$

Note that the change in $\mathbf{G}_2$ implies that at each point of the execution of the game and for each image $y \in \{0,1\}^\lambda$, there is at most one preimage $\mathsf{H}_t^{-1}(y)$ under H_t. By looking at the random oracle queries of $\mathcal{A}$, the game can extract preimages of given images y, and we know that for each y at most one preimage can be extracted. We will make use of such an extraction in the following games.

Game $\mathbf{G}_3$: We change the way the signer oracle is executed. In particular, when $\mathcal{A}$ sends $\mu_0, \mathsf{com}_\mathsf{r}, \mathsf{com}_c$ as its first message, the game tries to extract values $\bar{\mathsf{com}}_{\mathsf{r},i}$ such that $\mathsf{com}_\mathsf{r} = \mathsf{H}_\mathsf{r}(\bar{\mathsf{com}}_{\mathsf{r},1}, \ldots, \bar{\mathsf{com}}_{\mathsf{r},K})$ by searching through random oracle queries. If the game can not extract such a preimage, we write $\bar{\mathsf{com}}_{\mathsf{r},i} = \perp$ for all $i \in [K]$. Then, the game aborts if it can not extract such a preimage , i.e., $\bar{\mathsf{com}}_{\mathsf{r},i} = \perp$, but later algorithm Check outputs 1. Recall that algorithm Check verifies that

$$\mathsf{com}_\mathsf{r} = \mathsf{H}_\mathsf{r}(\mathsf{com}_{\mathsf{r},1}, \ldots, \mathsf{com}_{\mathsf{r},K}).$$

Thus, for every fixed interaction, we can bound the probability of such an abort by $Q_{\mathsf{H}_\mathsf{r}}/2^\lambda$. Indeed, once com_r is sent by $\mathcal{A}$ and thus fixed, and the game can not extract, we know that there is no bitstring x such that $\mathsf{H}_\mathsf{r}(x) = \mathsf{com}_\mathsf{r}$. Also, if algorithm Check outputs 1, we know that $\mathcal{A}$ was able to find a preimage of com_r after this was fixed. This can happen with probability at most $1/2^\lambda$ for each random oracle query. Using a union bound over all interactions we obtain

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{qQ_{\mathsf{H}_\mathsf{r}}}{2^\lambda}.$$

Game $\mathbf{G}_4$: We introduce another abort in the signer oracle. In this game, after the extraction of $(\bar{\mathsf{com}}_{\mathsf{r},1}, \ldots, \bar{\mathsf{com}}_{\mathsf{r},K})$ from com_r we introduced before, the game extracts $(\bar{\mathsf{r}}_{i,1}, \ldots, \bar{\mathsf{r}}_{i,N})$ from $\bar{\mathsf{com}}_{\mathsf{r},i}$ for every $i \in [K]$ for which $\bar{\mathsf{com}}_{\mathsf{r},i} \neq \perp$, such that

$$\bar{\mathsf{com}}_{\mathsf{r},i} = \mathsf{H}_\mathsf{r}(\mathsf{H}_\mathsf{r}(\bar{\mathsf{r}}_{i,1}), \ldots, \mathsf{H}_\mathsf{r}(\bar{\mathsf{r}}_{i,N})).$$

Again, the game does this by looking at the random oracle queries of $\mathcal{A}$ and we write $\bar{\mathsf{r}}_{i,j} = \perp$ if the game can not extract the value $\bar{\mathsf{r}}_{i,j}$. If there is an instance $i \in [K]$ and a session $j \in [N]$ such that $\bar{\mathsf{com}}_{\mathsf{r},i} \neq \perp$ but $\bar{\mathsf{r}}_{i,j} = \perp$ and later in that execution $\mathbf{J}_i \neq j$ but algorithm Check outputs 1, the game aborts.

To analyze the probability of this abort, fix an interaction and an instance $i \in [K]$. Assume that $\bar{\mathsf{com}}_{\mathsf{r},i} \neq \perp$ and there is a session $j \in [N]$ such that $\bar{\mathsf{r}}_{i,j} = \perp$ and later in that interaction $\mathbf{J}_i \neq j$. Then,

after $\bar{\mathsf{com}}_{\mathsf{r},i}$ is fixed, we consider two cases. In the first case, the game could not extract $h_1, \ldots, h_N$ such that $\bar{\mathsf{com}}_{\mathsf{r},i} = \mathsf{H}_{\mathsf{r}}(h_1, \ldots, h_N)$. Clearly, once $\bar{\mathsf{com}}_{\mathsf{r},i}$ is fixed, the probability that one of the hash queries of $\mathcal{A}$ evaluates to $\bar{\mathsf{com}}_{\mathsf{r},i}$ is at most $1/2^\lambda$. Thus, the probability that Check outputs 1, i.e., $\mathcal{A}$ is able to open $\bar{\mathsf{com}}_{\mathsf{r},i}$ in this case, is at most $Q_{\mathsf{H}_{\mathsf{r}}}/2^\lambda$. Similarly, in the case where the game could extract $h_1, \ldots, h_N$, but could not extract $\bar{\mathsf{r}}_{i,j}$ such that $\mathsf{H}_{\mathsf{r}}(\bar{\mathsf{r}}_{i,j}) = h_j$, the probability that one of $\mathcal{A}$'s hash queries evaluates to h_j is at most $1/2^\lambda$. Thus, the probability that Check outputs 1, i.e., $\mathcal{A}$ is able to open h_j in this case, is at most $Q_{\mathsf{H}_{\mathsf{r}}}/2^\lambda$. Note that here we needed that $j \neq \mathbf{J}_i$, as the definition of Check does not require $\mathcal{A}$ to open $h_{\mathbf{J}_i}$. Applying a union bound over the interactions and instances we get

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{qKQ_{\mathsf{H}_{\mathsf{r}}}}{2^\lambda}.$$

Game $\mathbf{G}_5$: We introduce another abort: whenever $\mathcal{A}$ sends $\mu_0, \mathsf{com}_{\mathsf{r}}, \mathsf{com}_c$ as its first message, the game behaves as before, but additionally the game extracts values $\bar{c}_{1,1}, \ldots, \bar{c}_{K,N}$ from com_c such that

$$\mathsf{com}_c = \mathsf{H}_c(\bar{c}_{1,1}, \ldots, \bar{c}_{K,N}).$$

If the game can not extract, but later algorithm Check outputs 1, the game aborts. Note that algorithm Check internally checks if

$$\mathsf{com}_c = \mathsf{H}_c(c_{1,1}, \ldots, c_{K,N}).$$

Thus, for each fixed interaction it is possible to argue as in the previous games to bound the probability of such an abort and hence we obtain

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{qQ_{\mathsf{H}_c}}{2^\lambda}.$$

Game $\mathbf{G}_6$: In $\mathbf{G}_6$, the signer oracle sends a random $\mathsf{com}_{\mathbf{J}}$ in the beginning of each interaction. Later, before it has to send $\mathsf{seed}_{\mathbf{J}}, \mathsf{salt}$, it samples $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ and aborts if $\mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, \mathsf{salt})$ is already defined. If it is not yet defined, it defines it as $\mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, \mathsf{salt}) := \mathsf{com}_{\mathbf{J}}$. The adversary $\mathcal{A}$ can only distinguish between $\mathbf{G}_5$ and $\mathbf{G}_6$ if $\mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, \mathsf{salt})$ is already defined. By a union bound over all $Q_{\mathsf{H}'}$ hash queries and q interactions we obtain

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{qQ_{\mathsf{H}'}}{2^\lambda}.$$

Game $\mathbf{G}_7$: In $\mathbf{G}_7$, the game aborts if in some interaction there exists an $i \in [K]$ such that $\mathsf{H}'(\mathsf{seed}_{\mathbf{J}}, i)$ has already been queried before the signing oracle sends $\mathsf{seed}_{\mathbf{J}}$ to $\mathcal{A}$. Clearly, $\mathcal{A}$ obtains no information about $\mathsf{seed}_{\mathbf{J}}$ before the potential abort, see $\mathbf{G}_6$. Further, $\mathsf{seed}_{\mathbf{J}}$ is sampled uniformly at random. A union bound over all $Q_{\mathsf{H}'}$ queries and q interactions shows that

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \frac{qQ_{\mathsf{H}'}}{2^\lambda}.$$

Now, fix an interaction in $\mathbf{G}_7$ and assume that Check returns 1 and the game does not abort due to any of the reasons we introduced so far. Note that this means that for all instances $i \in [K]$ the value $\bar{\mathsf{com}}_{\mathsf{r},i}$ could be extracted. Furthermore, this means that if there exists $i \in [K], j_0 \in [N]$ such that $\bar{\mathsf{r}}_{i,j_0} = \bot$ then later $\mathbf{J}_i = j_0$. Also, note that if Check does not abort, then we have $\bar{\mathsf{com}}_{\mathsf{r},i} = \mathsf{com}_{\mathsf{r},i}, \bar{\mathsf{r}}_{i,j} = \mathsf{r}_{i,j}$ and $\bar{c}_{i,j} = c_{i,j}$ for all $(i,j) \in [K] \times [N]$ for which these values are defined. This is because we ruled out collisions for oracles $\mathsf{H}_{\mathsf{r}}, \mathsf{H}_c$. Now, we define an indicator random variable $\mathsf{cheat}_{i,\mathsf{ctr}}$ for the event that in the ctr^{th} interaction, the signer oracle does not abort and there exists $i \in [K], j \in [N]$ such that $\bar{\mathsf{r}}_{i,j} = \bot$ or $\bar{\mathsf{r}}_{i,j} = (\alpha, \varphi, \gamma)$ such that

$$c_{i,j} \neq \mathsf{H}(\mathsf{pk}_i, \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi)) \cdot g^\alpha.$$

We say that $\mathcal{A}$ successfully cheats in instance $i \in [K]$ and interaction ctr if $\mathsf{cheat}_{i,\mathsf{ctr}} = 1$. We also define the number of interactions in which $\mathcal{A}$ successfully cheats in instance i as $\mathsf{cheat}_i^* := \sum_{\mathsf{ctr}=2}^{q+1} \mathsf{cheat}_{i,\mathsf{ctr}}$. By the discussion above, we have that $\mathsf{cheat}_{i,\mathsf{ctr}} = 1$ implies that $\mathbf{J}_i = j_0$ and thus

$$\Pr\left[\mathsf{cheat}_{i,\mathsf{ctr}} = 1\right] \leq \frac{1}{N}.$$

Therefore, we can bound the expectation of cheat_i^* using

$$\mathbb{E}\left[\mathsf{cheat}_i^*\right] \leq \frac{1}{3\vartheta \ln(q_{\max}+1)} \sum_{\mathsf{ctr}=2}^{q+1} \frac{1}{\mathsf{ctr}} \leq \frac{\ln(q+1)}{3\vartheta \ln(q_{\max}+1)} \leq \frac{1}{3\vartheta}.$$

Now, if we plug $X := \mathsf{cheat}_i^*$ and $s := 3\mathbb{E}\left[\mathsf{cheat}_i^*\right] + \delta = 1/\vartheta + \delta$ into the Chernoff bound (Lemma 3.1), we get that for all $i \in [K]$

$$\Pr\left[\mathsf{cheat}_i^* \geq \frac{1}{\vartheta} + \delta\right] \leq e^{-\delta}.$$

We note that the entire calculation of this probability also holds if we fix the random coins of the adversary.

Game $\mathbf{G}_8$: Game $\mathbf{G}_8$ is defined as $\mathbf{G}_7$, but additionally aborts if for all $i \in [K]$ we have $\mathsf{cheat}_i^* \geq \delta + 1/\vartheta$. In particular, if $\mathbf{G}_8$ does not abort, then there is some instance i for which $\mathcal{A}$ does not successfully cheat at all, which follows from the assumption $(1-\delta)\vartheta > 1$.

We can now bound the distinguishing advantage of $\mathcal{A}$ between $\mathbf{G}_7$ and $\mathbf{G}_8$ as follows. We denote the random coins of $\mathcal{A}$ by $\rho_{\mathcal{A}}$ and the random coins of the experiment (excluding $\rho_{\mathcal{A}}$) by ρ. Let bad be the event that for all $i \in [K]$ we have $\mathsf{cheat}_i^* \geq \delta + 1/\vartheta$. We note that the coins ρ that the experiment uses for the K instances are independent. Thus we have

$$\begin{aligned}
\Pr_{\rho,\rho_{\mathcal{A}}}\left[\mathsf{bad}\right] &= \sum_{\bar{\rho}_{\mathcal{A}}} \Pr_{\rho_{\mathcal{A}}}\left[\rho_{\mathcal{A}} = \bar{\rho}_{\mathcal{A}}\right] \cdot \Pr_{\rho,\rho_{\mathcal{A}}}\left[\mathsf{bad} \mid \rho_{\mathcal{A}} = \bar{\rho}_{\mathcal{A}}\right] \\
&= \sum_{\bar{\rho}_{\mathcal{A}}} \Pr_{\rho_{\mathcal{A}}}\left[\rho_{\mathcal{A}} = \bar{\rho}_{\mathcal{A}}\right] \cdot \prod_{i\in[K]} \Pr_{\rho,\rho_{\mathcal{A}}}\left[\mathsf{cheat}_i^* \geq \frac{1}{\vartheta} + \delta \;\middle|\; \rho_{\mathcal{A}} = \bar{\rho}_{\mathcal{A}}\right] \\
&\leq \sum_{\bar{\rho}_{\mathcal{A}}} \Pr_{\rho_{\mathcal{A}}}\left[\rho_{\mathcal{A}} = \bar{\rho}_{\mathcal{A}}\right] \cdot e^{-\delta K} = e^{-\delta K},
\end{aligned}$$

which implies

$$\left|\Pr\left[\mathbf{G}_7 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_8 \Rightarrow 1\right]\right| \leq \Pr_{\rho,\rho_{\mathcal{A}}}\left[\mathsf{bad}\right] \leq e^{-\delta K}.$$

Game $\mathbf{G}_9$: In game $\mathbf{G}_9$, we sample a random instance $i^* \xleftarrow{\$} [K]$ at the beginning of the game. In the end, the game aborts if $\mathsf{cheat}_{i^*}^* \geq \delta + 1/\vartheta$. In particular, if this game does not abort, then $\mathcal{A}$ does not successfully cheat in instance i^* at all. As $\mathcal{A}$'s view is independent from i^*, we have

$$\begin{aligned}
\Pr\left[\mathbf{G}_9 \Rightarrow 1\right] &= \Pr\left[\mathbf{G}_8 \Rightarrow 1 \wedge \mathsf{cheat}_{i^*}^* < \frac{1}{\vartheta} + \delta\right] \\
&= \Pr\left[\mathbf{G}_8 \Rightarrow 1\right] \cdot \Pr\left[\mathsf{cheat}_{i^*}^* < \frac{1}{\vartheta} + \delta \;\middle|\; \mathbf{G}_8 \Rightarrow 1\right] \\
&\geq \Pr\left[\mathbf{G}_8 \Rightarrow 1\right] \cdot \Pr\left[\mathsf{cheat}_{i^*}^* < \frac{1}{\vartheta} + \delta \;\middle|\; \exists i \in [K] : \mathsf{cheat}_i^* < \frac{1}{\vartheta} + \delta\right] \\
&\geq \Pr\left[\mathbf{G}_8 \Rightarrow 1\right] \cdot \frac{1}{K},
\end{aligned}$$

where the first inequality follows from the fact that the event $\mathbf{G}_8 \Rightarrow 1$ implies the event $\exists i \in [K] : \mathsf{cheat}_i^* < \delta + 1/\vartheta$. We note that from now on, our proof follows the proof strategy of the BLS signature scheme [BLS01].

Game $\mathbf{G}_{10}$: In game $\mathbf{G}_{10}$, we introduce an initially empty set $\mathcal{L}$ and a new abort. We highlight that we treat $\mathcal{L}$ as a set and therefore every bitstring is in $\mathcal{L}$ only once. Recall that when $\mathcal{A}$ sends $\mu_0, \mathsf{com}_\mathsf{r}, \mathsf{com}_c$ to the signer oracle, the game tries to extract values $\bar{\mathsf{r}}_{i,j}$ for $(i,j) \in [K] \times [N]$. Then the game samples $\mathsf{seed}_\mathbf{J}$ and computes $\mathbf{J}$ accordingly. In particular, due to the changes in the previous games we know that the game extracts $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\alpha, \varphi, \gamma)$ unless the experiment will abort anyways. Then, in game $\mathbf{G}_{10}$, the game will insert $\mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi)$ into $\mathcal{L}$ as soon as it needs to compute $s_{i^*,\mathbf{J}_{i^*}}$. This means that the list $\mathcal{L}$ contains at most as many entries as the number of completed interactions.

Fix the first pair (m, σ) in the adversary's final output such that for $\sigma = (\bar{\sigma}, \varphi_1, \ldots, \varphi_K)$ and $\mu^* := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_{i^*})$ we have $\mu^* \notin \mathcal{L}$. Such a pair must exists if $\mathcal{A}$ is successful, see game $\mathbf{G}_1$. Then game $\mathbf{G}_{10}$ aborts if $\mathsf{H}(\mathsf{pk}_{i^*}, \mu^*)$ is not defined yet. Note that $\mathcal{A}$'s success probability in such a case can be at most $1/p$ and hence

$$|\Pr[\mathbf{G}_9 \Rightarrow 1] - \Pr[\mathbf{G}_{10} \Rightarrow 1]| \leq \frac{1}{p}.$$

Game $\mathbf{G}_{11}$: In game $\mathbf{G}_{11}$, we change how the random oracle H is simulated and add a new abort. For every query of the form $\mathsf{H}(\mathsf{pk}_{i^*}, \mu)$ the game independently samples a bit $b[\mu] \in \{0,1\}$ such that the probability that $b[\mu] = 1$ is $1/(q+1)$. Whenever the game adds a value μ to the set $\mathcal{L}$, it aborts if $b[\mu] = 1$. Then, after $\mathcal{A}$ returns its final output, the game determines μ^* as in $\mathbf{G}_{10}$, adds arbitrary values to $\mathcal{L}$ such that all values in $\mathcal{L} \cup \{\mu^*\}$ are distinct and $|\mathcal{L}| = q$ and aborts if $b[\mu^*] = 0$ or there is a $\mu \in \mathcal{L}$ such that $b[\mu] = 1$. Otherwise it continues as before. Note that unless the game aborts, $\mathcal{A}$'s view does not change. As all bits $b[\mu]$ are independent, we derive

$$\begin{aligned}
\Pr[\mathbf{G}_{11} \Rightarrow 1] &= \Pr[\mathbf{G}_{10} \Rightarrow 1] \cdot \Pr[b[\mu^*] = 1 \wedge \forall \mu \in \mathcal{L} : b[\mu] = 0] \\
&= \Pr[\mathbf{G}_{10} \Rightarrow 1] \cdot \frac{1}{q+1} \left(1 - \frac{1}{q+1}\right)^q \\
&= \Pr[\mathbf{G}_{10} \Rightarrow 1] \cdot \frac{1}{q} \left(1 - \frac{1}{q+1}\right)^{q+1} \\
&\geq \Pr[\mathbf{G}_{10} \Rightarrow 1] \cdot \frac{1}{4q},
\end{aligned}$$

where the last inequality follows from $(1 - 1/x)^x \geq 1/4$ for all $x \geq 2$.

Finally, we construct a reduction $\mathcal{B}$ that solves CDH such that

$$\Pr[\mathbf{G}_{11} \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Then, the statement follows by an easy calculation. Reduction $\mathcal{B}$ works as follows:

- $\mathcal{B}$ gets as input bilinear group parameters $\mathbb{G}, g, p, e$ and group elements $X = g^x, Y = g^y$. The goal of $\mathcal{B}$ is to compute g^{xy}. First, $\mathcal{B}$ samples $i^* \xleftarrow{\$} [K]$. Then, it defines $\mathsf{pk}_{i^*} := X$ (which implicitly defines $\mathsf{sk}_{i^*} := x$) and $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{pk}_i := g^{\mathsf{sk}_i}$ for $i \in [K] \setminus \{i^*\}$.
- $\mathcal{B}$ runs adversary $\mathcal{A}$ on input $\mathbb{G}, g, p, e, \mathsf{pk} := (\mathsf{pk}_1, \ldots, \mathsf{pk}_K, \mathsf{ck})$ with oracle access to a signer oracle and random oracles $\mathsf{H}, \mathsf{H}_\mathsf{r}, \mathsf{H}_c, \mathsf{H}'$. To do so, it simulates oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_c, \mathsf{H}'$ exactly as in $\mathbf{G}_{11}$. The other oracles are provided as follows:
  - For a query of the form $\mathsf{H}(\mathsf{pk}_{i^*}, \mu)$ for which the hash value is not yet defined, it samples a bit $b[\mu] \in \{0,1\}$ such that the probability that $b[\mu] = 1$ is $1/(q+1)$. Then, it defines the hash value as $Y^{b[\mu]} \cdot g^{t[i^*,\mu]}$ for a randomly sampled $t[i^*, \mu] \xleftarrow{\$} \mathbb{Z}_p$. For a query of the form $\mathsf{H}(\mathsf{pk}_i, \mu), i \neq i^*$ for which the hash value is not yet defined it defines the hash value as $g^{t[i,\mu]}$ for a randomly sampled $t[i, \mu] \xleftarrow{\$} \mathbb{Z}_p$. For all other queries it simulates H honestly.

  - When $\mathcal{A}$ starts an interaction with the signer oracle, $\mathcal{B}$ sends N to $\mathcal{B}$ as in the protocol. When $\mathcal{B}$ sends its first message $\mu_0, \mathsf{com}_\mathsf{r}, \mathsf{com}_c$ as its first message, $\mathcal{B}$ behaves as $\mathbf{G}_{11}$. In particular, it tries to extract $\bar{\mathsf{r}}_{i,j}, \bar{c}_{i,j}$ for $(i,j) \in [K] \times [N]$. It then sends $\mathsf{seed}_\mathbf{J}$ to $\mathcal{A}$.
  - When $\mathcal{A}$ sends its second message $k_\mathbf{J}, \{c_{i,\mathbf{J}_i}, \eta_i\}_{i\in[K]}$, $\mathcal{B}$ aborts under the same conditions as $\mathbf{G}_{11}$ does. In particular, if $\mathcal{B}$ does not abort and the signer oracle does not abort then $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\alpha, \varphi, \gamma)$ is defined and $\mathcal{B}$ for $\mu := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi)$, $\mathcal{B}$ sets $s_{i^*,\mathbf{J}_{i^*}} := X^{t[i^*,\mu]+\alpha}$. As defined in $\mathbf{G}_{11}$, $\mathcal{B}$ also inserts μ into the set $\mathcal{L}$. It computes $s_{i,\mathbf{J}_i}$ for $i \neq i^*$ as game $\mathbf{G}_{11}$ does, which is possible as $\mathcal{B}$ holds the corresponding sk_i. Then, $\mathcal{B}$ sends $\bar{s} := \prod_{i=1}^{K} s_{i,\mathbf{J}_i}$ to $\mathcal{A}$.

- When $\mathcal{A}$ returns its final output, $\mathcal{B}$ performs all verification steps in $\mathbf{G}_{11}$. In particular, it searches for the first pair (m, σ) in $\mathcal{A}$'s final output such that for $\sigma = (\bar{\sigma}, \varphi_1, \ldots, \varphi_K)$ and $\mu^* := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_{i^*})$ we have $\mu^* \notin \mathcal{L}$. As defined in $\mathbf{G}_{11}$, $\mathcal{B}$ aborts if $b[\mu^*] = 0$. Finally, $\mathcal{B}$ defines $\mu_i := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_i)$ and returns

$$Z := \bar{\sigma} \cdot X^{-t[i^*,\mu^*]} \cdot g^{-\sum_{i\in[K]\setminus\{i^*\}} t[i,\mu_i]\mathsf{sk}_i}$$

  to its challenger.

We first argue that $\mathcal{B}$ perfectly simulates $\mathbf{G}_{11}$ for $\mathcal{A}$. To see that, note that as the $t[i,\mu]$ are sampled uniformly at random, the random oracle is simulated perfectly. To see that $s_{i^*,\mathbf{J}_{i^*}}$ is distributed correctly, note that if the signing oracle and $\mathbf{G}_{11}$ do not abort, then we have

$$c_{i^*,\mathbf{J}_{i^*}}^{\mathsf{sk}_{i^*}} = (\mathsf{H}(\mathsf{pk}_{i^*}, \mu) \cdot g^{\alpha})^{\mathsf{sk}_{i^*}} = \left(Y^{b[\mu]} \cdot g^{t[i^*,\mu]} \cdot g^{\alpha}\right)^{x} = X^{t[i^*,\mu]+\alpha},$$

where the last equality follows from $b[\mu] = 0$, as otherwise $\mathbf{G}_{11}$ would have aborted.

It remains to show that if $\mathbf{G}_{11}$ outputs 1, then we have $Z = g^{xy}$. This follows directly from the verification equation and $b[\mu^*] = 1$. To see this, note that

$$\begin{aligned}\prod_{i=1}^{K} e\left(\mathsf{H}(\mathsf{pk}_i, \mu_i), \mathsf{pk}_i\right) &= e\left(Y^{b[\mu^*]} \cdot g^{t[i^*,\mu^*]}, X\right) \cdot \prod_{i\in[K]\setminus\{i^*\}} e\left(g^{t[i,\mu_i]}, g^{\mathsf{sk}_i}\right) \\ &= e\left(g,g\right)^{xy+t[i^*,\mu^*]x} \cdot e\left(g,g\right)^{\sum_{i\in[K]\setminus\{i^*\}} t[i,\mu_i]\mathsf{sk}_i}.\end{aligned}$$

Using the verification equation, this implies that

$$g^{xy} = \bar{\sigma} \cdot g^{-\left(t[i^*,\mu^*]x + \sum_{i\in[K]\setminus\{i^*\}} t[i,\mu_i]\mathsf{sk}_i\right)}$$

Concluded. □

We note that instead of giving games $\mathbf{G}_{10}, \mathbf{G}_{11}$ and the reduction from CDH explicitly, one can also directly reduce from the security of the BLS signature scheme to $\mathbf{G}_9$, leading to the very same bound in total. This tells us that one can use (up to losing $\log(K)$ bits[6] of security) the same curves as for BLS. We summarize this observation in the following lemma.

Lemma 3.4 (Informal). *Under the same conditions as in Theorem 3.2, the scheme* $\mathsf{BS}_{\mathsf{CDH}}$ *satisfies* $q_{\max}$*-one-more unforgeability, if the BLS signature scheme [BLS01] is unforgeable under chosen message attacks relative to* PGGen*, where the concrete security loss is (up to statistically negligible terms) given by* K*.*

[6] In our concrete instantiation, $\log(K) \approx 6.5$.

3.4.2 Concrete Parameters and Efficiency

Let us now discuss concrete parameters for our scheme $\mathsf{BS}_{\mathsf{CDH}}$ based on the CDH assumption. Recall that the scheme uses parameters K, ϑ, and p. Instantiating the commitment scheme CMT with a Pedersen commitment we also have to set a value for the order p' of the group that is used in this commitment scheme. Say that we aim for κ bits of security. In particular, we want to find appropriate values for $K, \vartheta, |p|$ and $|p'|$. Consider an adversary with running time t and advantage ϵ against the OMUF security of the scheme. If $\epsilon/t < 2^{-\kappa}$, we are done. Otherwise, we have $\epsilon/t \geq 2^{-\kappa}$ and $\epsilon \geq 2^{-\kappa}$, as $t \geq 1$. Now, we want to use Theorem 3.2 to end up with a contradiction. We assume κ_{CDH} bits of security for the CDH instance and κ_{CMT} bits of security for the commitment scheme CMT. If we use Theorem 3.2 with $\delta := -\ln(\epsilon/2)/K$, then $e^{-\delta K} = \epsilon/2$ and the security bound becomes

$$\epsilon \leq 2\left(2^{-\kappa_{\mathsf{CMT}}} \cdot t + \frac{K}{p} + 4qK \cdot 2^{-\kappa_{\mathsf{CDH}}} \cdot t + \mathsf{stat}\right).$$

We can now increase κ_{CDH} and κ_{CMT} (for a fixed combination of ϵ and t) until this inequality does not hold anymore. Then the adversary could not have existed in the first place. Using κ_{CDH} and κ_{CMT}, we can then determine an appropriate choice for $|p| = 2\kappa_{\mathsf{CDH}} + 1$ and $|p'| = 2\kappa_{\mathsf{CMT}} + 1$, see [Pol78].

However, note that we can only apply this approach, if $(1-\delta)\vartheta > 1$, due to Theorem 3.2. By our choice of δ it is therefore sufficient to guarantee that $\left(1 - \ln(2^{\kappa+1})/K\right)\vartheta > 1$. It is clear that for a decreasing K, we have to increase ϑ to satisfy this constraint. Thus, our approach is as follows: For a few choices of K, we determine the minimum $\vartheta > 0$, such that the constraint holds. If there is no such ϑ, we throw away this particular K. Then, we proceed as discussed above to find security levels for the underlying instances and compute the signature sizes and key sizes[7].

Next, we focus on blindness. For simplicity, assume that $N^L = N^R =: N$. We instantiate PRF using a GGM construction with a random oracle $\mathsf{H}_{\mathsf{PRF}}$ (cf. Section 3.3) and know that $\epsilon_{\mathsf{PRF}} \leq (2\log(NK) - 1)KQ_{\mathsf{H}_{\mathsf{PRF}}}/2^{\lambda_{\mathsf{PRF}}}$, where λ_{PRF} is the output length of the pseudorandom function. By applying Theorem 3.1 we obtain the security bound

$$\frac{(2\log(N) + 2\log(K) - 1)\,KQ_{\mathsf{H}_{\mathsf{PRF}}}}{2^{\lambda_{\mathsf{PRF}}-2}} + \frac{Q_{\mathsf{H}'}^2}{2^{\lambda-1}} + \frac{Q_{\mathsf{H}'}}{2^{\lambda-2}} + \frac{Q_{\mathsf{H}_x}}{2^{\lambda_{\mathsf{PRF}}-2}} + \frac{KQ_{\mathsf{H}_r}}{2^{\lambda_{\mathsf{PRF}}-2}},$$

where λ_{PRF} denotes the output length of PRF. Thus, we only have to increase λ_{PRF} until the security bound guarantees κ bit of security.

We implemented the approach discussed above in Python script[8]. To simplify a bit, we made the conservative assumption that the number of hash queries for each random oracle is equal to the running time of the adversary and set N^L and N^R in the blindness bound to be equal to the maximum number q of signatures interactions that the adversary starts. Results can be found in Table 3.1.

3.5 PI-Cut-Choo's Friend from RSA

In this section, we present our blind signature scheme based on the RSA assumption. Before we present the construction, we recall some background, namely, the Okamoto-Guillou-Quisquater [Oka93] function, and the boosting transform from [KLR21] instantiated with this function.

3.5.1 The OGQ Linear Function

Our scheme is based on the Okamoto-Guillou-Quisquater (OGQ) [Oka93] linear function. The function is specified by public parameters $\mathsf{par} = (n, a, \gamma)$ where p and q are distinct large primes and $n = pq$,

[7] In practice, we would use an asymmetric type-3 pairing for efficiency. This means that the public key has to be given in both source groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Our concrete parameter calculations take this into account.

[8] The Python script can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.

$a \xleftarrow{\$} \mathbb{Z}_n^*$ is sampled uniformly at random, and γ is a prime with $\gcd(n, \gamma) = \gcd(\varphi(n), \gamma) = 1$. Further, define a trapdoor $\mathsf{td} := (p, q)$. Throughout this section, we assume that these parameters are output by algorithm RSAGen. Let $\mathcal{D} := \mathbb{Z}_\gamma \times \mathbb{Z}_n^*$. It can be shown [HKL19] that $\mathcal{D}$ forms a group with respect to the group operation

$$(x_1, y_1) \circ (x_2, y_2) := \left(x_1 + x_2 \bmod \gamma, y_1 \cdot y_2 \cdot a^{\lfloor \frac{x_1 + x_2}{\gamma} \rfloor} \bmod n\right).$$

We specify a linear function F as follows:

$$\mathsf{F} \colon \mathcal{D} \to \mathbb{Z}_n^*, \;\; (x, y) \mapsto a^x y^\gamma \bmod n.$$

In addition, we specify a function

$$\Psi \colon \mathbb{Z}_n^* \times \mathbb{Z}_\gamma \times \mathbb{Z}_\gamma \to \mathcal{D}, \;\; (x, s, s') \mapsto (0, x^{\lfloor -\frac{s+s'}{\gamma} \rfloor} \bmod n).$$

These funcions satisfy

$$\begin{aligned} \forall x, y \in \mathcal{D}, s \in \mathbb{Z}_\gamma : \mathsf{F}(x^s \circ y) = \mathsf{F}(x)^s \cdot \mathsf{F}(y), \\ \forall y \in \mathbb{Z}_n^*, s, s' \in \mathbb{Z}_\gamma : y^{s+s'} = y^s \cdot y^{s'} \cdot \mathsf{F}(\Psi(y, s, s')). \end{aligned}$$

The collision resistance and one-wayness of the function F is tightly implied by the RSA assumption. For more details, see [HKL19]. We argue that the trapdoor can be used to sample uniform preimages for F. To this end, we specify an algorithm $\mathsf{Invert}(\mathsf{td}, z)$ for $z \in \mathbb{Z}_n^*$, which works as follows:

- Use p and q to compute $\rho \in \mathbb{Z}$ such that $\rho\gamma \bmod \varphi(n) = 1$.
- Sample $x \xleftarrow{\$} \mathbb{Z}_\gamma$ and set $y := (za^{-x})^\rho \bmod n$. Return (x, y).

In the following, we argue that Invert outputs properly distributed preimages. It is clear that for $(x, y) \leftarrow \mathsf{Invert}(\mathsf{td}, z)$, we have

$$\mathsf{F}(x, y) = a^x y^\gamma = a^x (za^{-x})^{\gamma\rho \bmod \varphi(n)} = z \pmod n.$$

Thus, it remains to show that the distributions

$$\mathcal{D}_1 := \{((x, y), z) \mid (x, y) \xleftarrow{\$} \mathbb{Z}_\gamma \times \mathbb{Z}_n^*, z := a^x y^\gamma \bmod n\}$$

and

$$\mathcal{D}_2 := \left\{((x, y), z) \;\middle|\; z \xleftarrow{\$} \mathbb{Z}_n^*, x \xleftarrow{\$} \mathbb{Z}_\gamma, y := (za^{-x})^\rho \bmod n\right\}$$

are the same. Fix $(x_0, y_0, z_0) \in \mathbb{Z}_\gamma \times \mathbb{Z}_n^* \times \mathbb{Z}_n^*$. As a is invertible and $y \mapsto y^\gamma$ defines a permutation on $\mathbb{Z}_n^*$, we have

$$\Pr_{(x,y,z) \leftarrow \mathcal{D}_1}[z = z_0] = \frac{1}{\varphi(n)} = \Pr_{(x,y,z) \leftarrow \mathcal{D}_2}[z = z_0].$$

By conditioning on $z = z_0$ we see that it remains to show that

$$\Pr_{(x,y,z) \leftarrow \mathcal{D}_1}[(x, y) = (x_0, y_0) \mid z = z_0] = \Pr_{(x,y,z) \leftarrow \mathcal{D}_2}[(x, y) = (x_0, y_0) \mid z = z_0].$$

Here, the left-hand side is equal to $1/\gamma$ if $z_0 = a^{x_0} y_0^\gamma \bmod n$ and 0 otherwise. The right-hand side is equal to $1/\gamma$ if $y_0 = (z_0 a^{-x_0})^\rho \bmod n$ and 0 otherwise. As both conditions are equivalent, we can conclude the analysis.

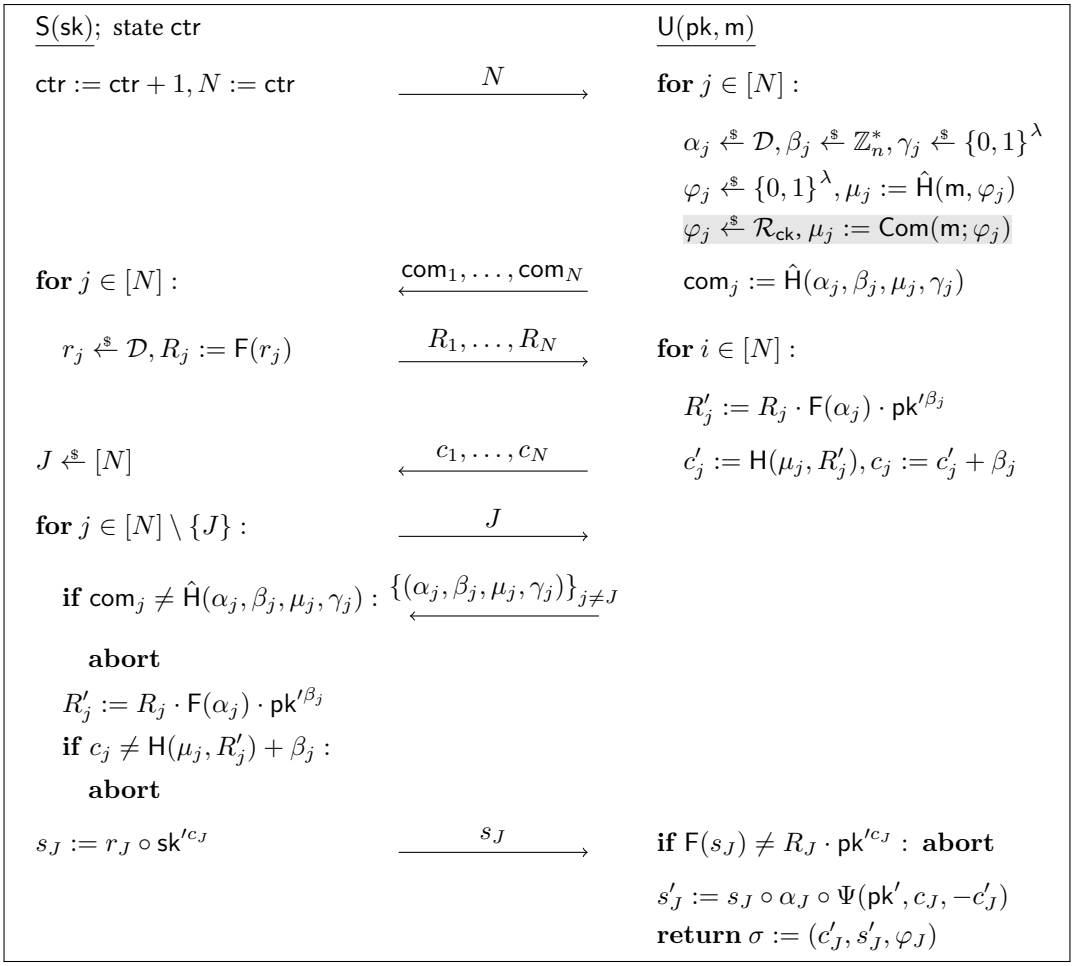

Figure 3.3: The signature issuing protocol of the blind signature scheme obtained via the boosting construction applied to the OGQ function. Here, $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$, $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}^\lambda$ are random oracles. The state ctr of S is atomically incremented at the beginning of every interaction. Instead of generating the commitments μ_i via a random oracle, we can also generate it via a commitment scheme (highlighted line). As long as it is binding, one can easily verify that the proof goes through.

3.5.2 The Underlying Boosting Transform

We revisit the boosting transform introduced in [KLR21] for the special case of the OGQ linear function. The boosting transform defines a blind signature scheme CCBS as follows.

Key Generation. Algorithm $\mathsf{CCBS.Gen}(1^\lambda)$ generates keys as:

1. Generate parameters $\mathsf{par} = (n, a, \gamma)$ as above.
2. Sample $\mathsf{sk}' \xleftarrow{\$} \mathcal{D}$, set $\mathsf{pk}' := \mathsf{F}(\mathsf{sk}')$.
3. Return the public key $\mathsf{pk} := (\mathsf{par}, \mathsf{pk}')$ and the secret key $\mathsf{sk} := \mathsf{sk}'$.

Signature Issuing. The signature issuing protocol of the scheme is presented in Figure 3.3. Here, the Signer is stateful and its state ctr is intitialized as $\mathsf{ctr} := 1$.

Verification. A signature $\sigma = (c', s', \varphi^*)$ is verified with respect to a message m via algorithm $\mathsf{CCBS.Ver}(\mathsf{pk} = (\mathsf{par}, \mathsf{pk}'), \mathsf{m}, \sigma)$, which is as follows:

1. Compute the commitment $\mu^* := \hat{\mathsf{H}}(\mathsf{m}, \varphi^*)$
2. Return 1 if $c' = \mathsf{H}(\mu^*, \mathsf{F}(s') \cdot \mathsf{pk}'^{-c'})$. Otherwise return 0.

We highlight that for the proof of one-more unforgeability in [KLR21] it is not important that the commitments μ_i are computed using a random oracle. In fact, what it needed is only that this commitment is binding. Indeed, it is easy to see that the proof goes through using any binding commitment scheme. We denote this modified scheme using a commitment scheme CMT by $\mathsf{CCBS}[\mathsf{CMT}]$. We summarize the one-more unforgeability bounds of the scheme $\mathsf{CCBS}[\mathsf{CMT}]$ in the following theorem. The concrete bounds can easily be derived from [HKL19, KLR21].

Theorem 3.3. *Let* CMT *be a randomness homomorphic commitment scheme. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$ *and* $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}^\lambda$ *be random oracles. Then,* $\mathsf{BS} := \mathsf{CCBS}[\mathsf{CMT}]$ *satisfies one-more unforgeability, if the* RSA *assumption holds relative to* RSAGen.

Precisely, for every $L = \mathsf{poly}(\lambda)$ *and every PPT algorithm* $\mathcal{A}$ *that makes at most* $Q_\mathsf{H}, Q_{\hat{\mathsf{H}}}$ *queries to oracles* $\mathsf{H}, \hat{\mathsf{H}}$*, respectively, and starts at most* p *interactions with his signer oracle, there exist PPT algorithms* $\mathcal{B}, \mathcal{B}', \mathcal{B}''$ *with* $\mathbf{T}(\mathcal{B}) \approx 2\mathbf{T}(\mathcal{A})$, $\mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{B}'') \approx \mathbf{T}(\mathcal{A})$, *and*

$$\mathsf{Adv}^{\mathsf{RSA}}_{\mathcal{B},\mathsf{RSAGen}}(\lambda) \geq \frac{1}{Q_\mathsf{H}^2 \ell^3} \left(\frac{\mathsf{Adv}^{L\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda)}{4} - \frac{\mathsf{Adv}^{\mathsf{bind}}_{\mathcal{B}'',\mathsf{CMT}}(\lambda) + p \cdot \mathsf{Adv}^{\mathsf{RSA}}_{\mathcal{B}',\mathsf{RSAGen}}(\lambda)}{2} - \frac{\mathsf{stat}}{2} - \frac{\mathsf{ex}}{\gamma} \right)^3,$$

where $\mathsf{stat} = \left(Q_{\hat{\mathsf{H}}}^2 + pQ_{\hat{\mathsf{H}}} + p^4 + p^2 Q_\mathsf{H}\right)/2^\lambda$, $\mathsf{ex} = (Q_\mathsf{H}(p-\ell))^{\ell+1}$, *and*

$$\ell = 3\ln(p+1) + \ln\left(2/\mathsf{Adv}^{L\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda)\right).$$

3.5.3 Construction

Now, we are ready to present our blind signature scheme BS_RSA, which makes use of a randomness homomorphic commitment scheme CMT with randomness space $\mathcal{R}_\mathsf{ck}$ and a puncturable pseudorandom function PRF. To recall, we can instantiate PRF using random oracles and CMT tightly based on the RSA assumption. Furthermore, we need random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$, $\mathsf{H}'\colon \{0,1\}^* \to \mathbb{Z}_n^*$, $\mathsf{H}''\colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda$, $\mathsf{H}_x\colon \{0,1\}^* \to \mathcal{D} \times \mathbb{Z}_\gamma \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$, where λ_PRF is a security parameter used for PRF.

Key Generation. Algorithm $\mathsf{BS}_\mathsf{RSA}.\mathsf{Gen}(1^\lambda)$ generates keys as follows:

1. Generate parameters $\mathsf{par} = (n, a, \gamma)$ and a trapdoor $\mathsf{td} = (p, q)$ as above (see Section 3.5.1).
2. Sample $\mathsf{sk}' \xleftarrow{\$} \mathcal{D}$, set $\mathsf{pk}' := \mathsf{F}(\mathsf{sk}')$.
3. Generate a commitment key $\mathsf{ck} \leftarrow \mathsf{CMT.Gen}(1^\lambda)$.
4. Set the state of S to $\mathsf{ctr} := 1$.
5. Return the public key $\mathsf{pk} := (\mathsf{par}, \mathsf{pk}', \mathsf{ck})$ and the secret key $\mathsf{sk} := (\mathsf{td}, \mathsf{sk}')$.

Signature Issuing. The algorithms S, U of the signature issuing protocol are formally presented in Figures 3.4 and 3.5. We note that S keeps a state ctr, which is inititalized as $\mathsf{ctr} := 1$.

Verification. A signature $\sigma = (c', s', \varphi^*)$ is verified with respect to a message m via algorithm $\mathsf{BS}_\mathsf{RSA}.\mathsf{Ver}(\mathsf{pk} = (\mathsf{par}, \mathsf{pk}', \mathsf{ck}), \mathsf{m}, \sigma)$, which is as follows:

1. Compute the commitment $\mu^* := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi^*)$.
2. Return 1 if $c' = \mathsf{H}(\mu^*, \mathsf{F}(s') \cdot \mathsf{pk}'^{-c'})$. Otherwise return 0.

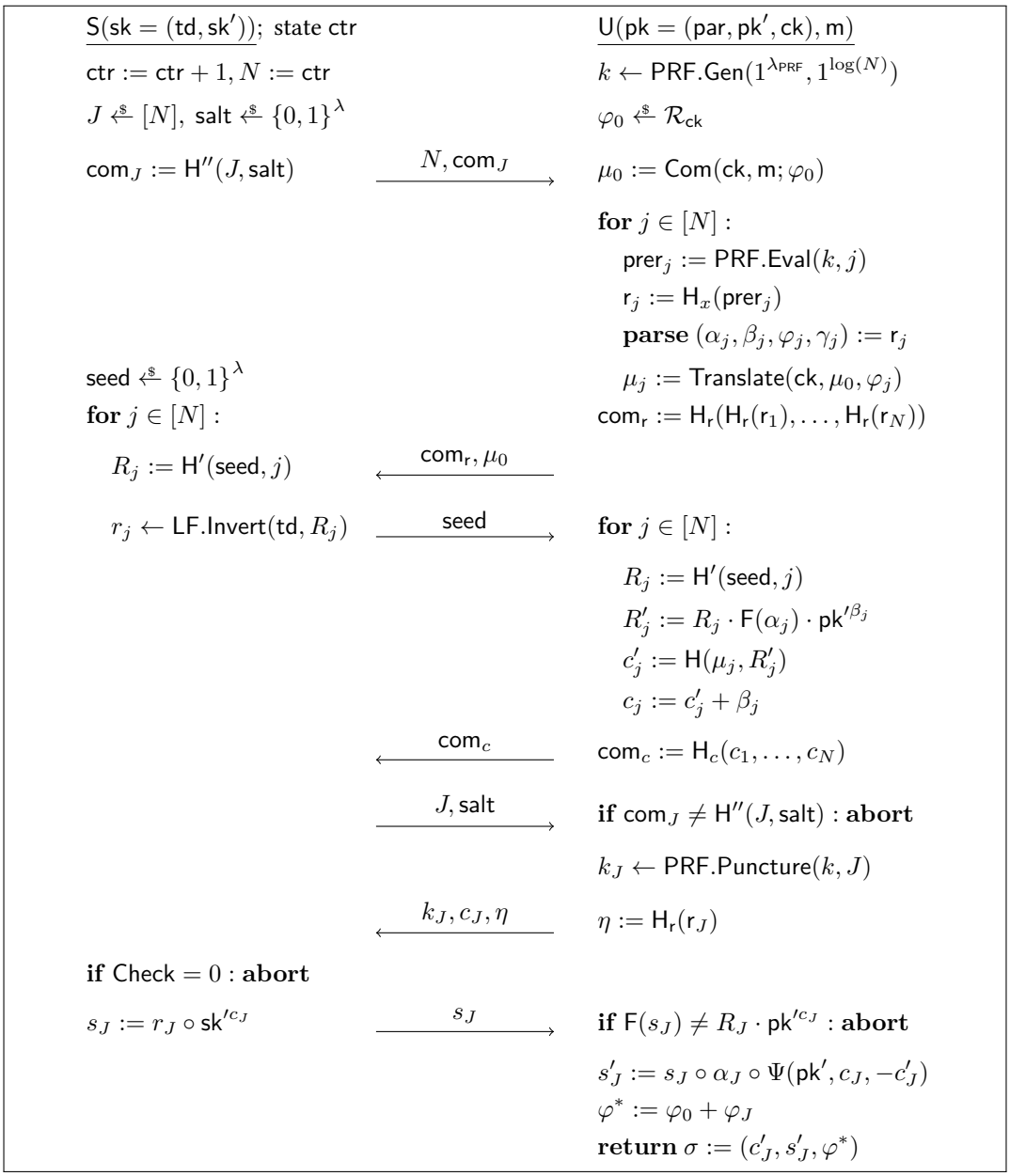

Figure 3.4: The signature issuing protocol of the blind signature scheme $\mathsf{BS}_{\mathsf{RSA}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$, $\mathsf{H}'\colon \{0,1\}^* \to \mathbb{Z}_n^*$, $\mathsf{H}''\colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda$, $\mathsf{H}_x\colon \{0,1\}^* \to \mathcal{D} \times \mathbb{Z}_\gamma \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$ are random oracles. The algorithm Check is defined in Figure 3.5. The state ctr of S is atomically incremented at the beginning of every interaction.

Alg $\mathsf{Check}(\mathsf{pk}, N, \mathsf{seed}, \mu_0, \mathsf{com}_\mathsf{r}, \mathsf{com}_c, J, k_J, c_J, \eta)$

```
01 for j ∈ [N] \ {J} :
02    prer_j := PRF.Eval(k_J, j),  r_j := H_x(prer_j)
03    parse (α_j, β_j, φ_j, γ_j) := r_j,  (α_j, β_j, φ_j, γ_j) ∈ D × Z_γ × R_ck × {0,1}^λ_PRF
04    R_j := H'(seed, j),  μ_j := Translate(ck, μ_0, φ_j),  c_j := H(μ_j, R_j · F(α_j) · pk'^β_j) + β_j
05 if com_r ≠ H_r(H_r(r_1), ..., H_r(r_{J-1}), η, H_r(r_{J+1}), ..., H_r(r_N)) : return 0
06 if com_c ≠ H_c(c_1, ..., c_N) : return 0
07 return 1
```

Figure 3.5: Algorithm Check used in the issuing protocol of blind signature scheme $\mathsf{BS}_{\mathsf{RSA}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$, $\mathsf{H}'\colon \{0,1\}^* \to \mathbb{Z}_n^*$ and $\mathsf{H}_\mathsf{r}, \mathsf{H}_c\colon \{0,1\}^* \to \{0,1\}^\lambda$, $\mathsf{H}_x\colon \{0,1\}^* \to \mathcal{D} \times \mathbb{Z}_\gamma \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$ are random oracles.

Analysis. Completeness of the scheme can easily be verified. We show blindness and one-more unforgeability. For blindness, we will use the following lemma. Informally, it states that even with a trapdoor, the linear blind signature scheme from the OGQ linear function satisfies blindness. The proof is an easy extension of the proof in [HKL19]. For details, see Theorem B.3 in [CAHL+22b].

Lemma 3.5. *For any algorithm $\mathcal{A}$ and bit $b \in \{0,1\}$, we consider the following game $\boldsymbol{G}_b$:*

1. *Let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$ be a random oracle. Run $(\rho, \mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}^\mathsf{H}(1^\lambda)$. Use ρ as random coins to compute $n, p, q, \gamma, a, \mathsf{sk}', \mathsf{pk}'$ as in the key generation of BS_RSA.*
2. *Let $O_{b'}$ for $b' \in \{0,1\}$ be an interactive oracle. Upon termination, it locally outputs $\sigma_{b\oplus b'}$ to the game. The oracle is defined as follows:*
   (a) *Receive R from $\mathcal{A}$, sample $(\alpha, \beta) \xleftarrow{\$} \mathcal{D} \times \mathbb{Z}_\gamma$, set $R' := R \cdot \mathsf{F}(\alpha) \cdot \mathsf{pk}'^\beta$. Set $c' := \mathsf{H}(\mathsf{m}_{b\oplus b'}, R')$ and $c := c' + \beta$. Send c to $\mathcal{A}$.*
   (b) *Receive s from $\mathcal{A}$. If $\mathsf{F}(s) \neq R \cdot \mathsf{pk}'^c$, define the local output of this oracle to be $\sigma_{b\oplus b'} := \bot$. Otherwise, set $s' := s \circ \alpha \circ \Psi(\mathsf{pk}', c, -c')$ and define the local output of this oracle to be $\sigma_{b\oplus b'} := (c', s')$.*
3. *Run $\mathcal{A}$ on input St with arbitrary interleaved one-time access to each of these oracles, i.e.,*
$$St' \leftarrow \mathcal{A}^{O_0, O_1, \mathsf{H}}(St).$$
4. *If $\sigma_0 = \bot$ or $\sigma_1 = \bot$, run $b^* \leftarrow \mathcal{A}(St', \bot, \bot)$. Else, run $b^* \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$. Output b^*.*

Then, for each algorithm $\mathcal{A}$ that makes at most Q_H many queries to H we have

$$|\Pr[\boldsymbol{G}_0 \Rightarrow 1] - \Pr[\boldsymbol{G}_1 \Rightarrow 1]| \leq \frac{4Q_\mathsf{H}}{|\mathbb{Z}_n^*|}.$$

Theorem 3.4. *Let PRF be a puncturable pseudorandom function, CMT be a randomness homomorphic commitment scheme. Further, let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_\gamma$, $\mathsf{H}''\colon \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathsf{H}_\mathsf{r}\colon \{0,1\}^* \to \{0,1\}^\lambda$, $\mathsf{H}_x\colon \{0,1\}^* \to \mathcal{D} \times \mathbb{Z}_\gamma \times \mathcal{R}_\mathsf{ck} \times \{0,1\}^{\lambda_\mathsf{PRF}}$ be random oracles. Then, BS_RSA satisfies semi-honest signer blindness.*

In particular, for any algorithm $\mathcal{A}$ that uses N^L and N^R as the counters in its interactions with the User and queries $\mathsf{H}, \mathsf{H}_\mathsf{r}, \mathsf{H}_x, \mathsf{H}''$ at most $Q_\mathsf{H}, Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_x}, Q_{\mathsf{H}''}$ times, respectively, there is an algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}_\mathsf{RSA}}(\lambda) \leq 4 \cdot \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda) + \frac{Q^2_{\mathsf{H}''}}{2^{\lambda-1}} + \frac{Q_{\mathsf{H}''}}{2^{\lambda-2}} + \frac{Q_{\mathsf{H}_x}}{2^{\lambda_\mathsf{PRF}-2}} + \frac{Q_{\mathsf{H}_\mathsf{r}}}{2^{\lambda_\mathsf{PRF}-2}} + \frac{4Q_\mathsf{H}}{|\mathbb{Z}_n^*|},$$

where $\mathcal{B}$ punctures PRF at one point with input length $d = \max\{\log(N^L), \log(N^R)\}$.

Proof. Set $\mathsf{BS} := \mathsf{BS}_{\mathsf{RSA}}$. Let $\mathcal{A}$ be a PPT adversary against blindness. We will show the statement via a sequence of games. Unless otherwise stated, random oracles are simulated honestly via lazy sampling.

Game $\mathbf{G}_{0,b}$: Game $\mathbf{G}_{0,b}$ is defined as the real blindness game $\mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}$. Recall that the game first obtains random coins ρ and two messages $\mathsf{m}_0, \mathsf{m}_1$ from the adversary $\mathcal{A}$. It computes a public key pk as the output of Gen on random coins ρ. Afterwards, $\mathcal{A}$ will interact with two oracles O_0 and O_1, simulating $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b)$ and $\mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$, respectively. We will reference to the variables used in these executions using superscripts L and R, respectively. For example, J^L refers to the index J sent by $\mathcal{A}$ in the interaction with oracle O_0. If we omit the superscript, we mean that our description applies to both oracles. According to this, N^L and N^R denote the cut-and-choose parameters sent by $\mathcal{A}$ in the first message of the interaction with oracles $\mathrm{O}_0, \mathrm{O}_1$, respectively. By definition, we have

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}_{\mathsf{RSA}}}(\lambda) = \left|\Pr\left[\mathbf{G}_{0,0} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{0,1} \Rightarrow 1\right]\right|.$$

Game $\mathbf{G}_{1,b}$: Game $\mathbf{G}_{1,b}$ is exactly as game $\mathbf{G}_{0,b}$, except that it aborts whenever there is a collision for random oracle H''. That is, whenever there are queries $\mathsf{H}''(x) = \mathsf{H}''(x')$ for $x \neq x'$. Clearly, the distinguishing advantage between games $\mathbf{G}_{1,b}$ and $\mathbf{G}_{0,b}$ can be bounded by the probability of such a collision, which leads to

$$\left|\Pr\left[\mathbf{G}_{0,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right]\right| \leq \frac{Q^2_{\mathsf{H}''}}{2^\lambda}.$$

Game $\mathbf{G}_{2,b}$: Game $\mathbf{G}_{2,b}$ is exactly as game $\mathbf{G}_{1,b}$, except that we introduce another abort. In this game, whenever the adversary sends N, com_J as its first message, the game searches for a query $\mathsf{H}''(\hat{J}, \hat{\mathsf{salt}}) = \mathsf{com}_J$. Note that the game can find at most one such query due to the previous change. If the game does not find such a query, but later the User does not abort, as the adversary successfully opens $\mathsf{com}_{\mathbf{J}}$ by sending J, salt, the game aborts. It is easy to see that the probability of this event is at most $Q_{\mathsf{H}''}/2^\lambda$ for fixed com_J and thus a union bound over $\{L, R\}$ leads to

$$\left|\Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{2,b} \Rightarrow 1\right]\right| \leq \frac{Q_{\mathsf{H}''}}{2^{\lambda-1}}.$$

Note that from now on, we can focus on the case where the game is able to find the query $\mathsf{H}''(\hat{J}, \hat{\mathsf{salt}}) = \mathsf{com}_J$, as otherwise the user oracle does abort. In particular, this implies that $\hat{J} = J$. If the user oracle does abort, the adversary does not learn anything about the bit b as CMT is perfectly hiding and no information about the randomness φ_0 is ever revealed to $\mathcal{A}$. For the rest of the proof, $\hat{J}$ denotes the cut-and-choose index that is extracted by the game from the commitment com_J and J is the cut-and-choose index that is later sent by $\mathcal{A}$ to open com_J. As said, we focus on the case where these are equal.

Game $\mathbf{G}_{3,b}$: Game $\mathbf{G}_{3,b}$ is defined exactly as $\mathbf{G}_{2,b}$, except that we change the way the randomness seeds prer_j are generated. Recall that before, these values were generated as in the real scheme, i.e.,

$$\mathsf{prer}_j := \mathsf{PRF.Eval}(k, j) \text{ for all } j \in [N].$$

Instead, we now generate these values using a punctured key $k_{\hat{J}}$ for $j \neq \hat{J}$, and as before for $j = \hat{J}$. To be precise, at the beginning of the interaction, we sample $k \leftarrow \mathsf{PRF.Gen}(1^{\lambda_{\mathsf{PRF}}}, 1^{\log(N)})$ as before, but additionally generate $k_{\hat{J}} \leftarrow \mathsf{PRF.Puncture}(k, \hat{J})$. Then we sample

$$\mathsf{prer}_{\hat{J}} := \mathsf{PRF.Eval}(k, \hat{J})$$

and

$$\mathsf{prer}_j := \mathsf{PRF.Eval}(k_{\hat{J}}, j) \text{ for all } j \in [N] \setminus \{\hat{J}\}.$$

Clearly, by the completeness of PRF this is only a syntactical change, and hence

$$\Pr\left[\mathbf{G}_{3,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{2,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{4,b}$: In game $\mathbf{G}_{4,b}$, we change the way we generate the values $\mathsf{prer}^L_{\hat{J}^L}$. Namely, we sample $\mathsf{prer}^L_{\hat{J}^L} \xleftarrow{\$} \{0,1\}^{\lambda}_{\mathsf{PRF}}$. The difference between $\mathbf{G}_{3,b}$ and $\mathbf{G}_{4,b}$ can now be bounded using the security of the puncturable pseudorandom function PRF. To be precise, we construct a reduction $\mathcal{B}$ as follows:

- Run $(\rho, \mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}(1^\lambda)$ and set $(\mathsf{pk}, \mathsf{sk}) := \mathsf{Gen}(1^\lambda; \rho)$.
- Run $\mathcal{A}$ on input St with access to random oracles and interactive oracles $\mathrm{O}_0, \mathrm{O}_1$, i.e., $St' \leftarrow \mathcal{A}^{\mathrm{O}_0,\mathrm{O}_1}(St)$. The oracle O_1 is provided as in game $\mathbf{G}_{3,b}$ and oracle O_0 is provided as follows:
  - When $\mathcal{A}$ sends N^L, com^L_J, extract $\hat{J}^L$ from com^L_J as game $\mathbf{G}_{3,b}$ does and output $\hat{J}^L$ to the PRF challenger. Obtain the punctured key $k_{\hat{J}^L}$ and value $\mathsf{prer}^L_{\hat{J}^L}$.
  - Use $k_{\hat{J}^L}$ to sample prer^L_j for $j \in [N^L] \setminus \{\hat{J}^L\}$ as in $\mathbf{G}_{3,b}$. Continue the oracle simulation as in $\mathbf{G}_{3,b}$. According to this, if the simulation does not abort, send the key $k_{\hat{J}^L}$ in the sixth message of the interaction.
- Let σ_b, σ_{1-b} be the local outputs of $\mathrm{O}_0, \mathrm{O}_1$, respectively. If $\sigma_0 = \bot$ or $\sigma_1 = \bot$, then run $b' \leftarrow \mathcal{A}(St', \bot, \bot)$. Else, run $b' \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$ and output b'.

Note that if $\mathsf{prer}^L_{\hat{J}^L_i}$ is random, then $\mathcal{B}$ perfectly simulates $\mathbf{G}_{4,b}$, whereas if $\mathsf{prer}^L_{\hat{J}^L_i} = \mathsf{Eval}(k, \hat{J})$, $\mathcal{B}$ perfectly simulates $\mathbf{G}_{3,b}$. By the security of PRF with input length $\log(N^L)$ we obtain

$$|\Pr[\mathbf{G}_{3,b} \Rightarrow 1] - \Pr[\mathbf{G}_{4,b} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda).$$

Game $\mathbf{G}_{5,b}$: In game $\mathbf{G}_{5,b}$, we change the way we generate $\mathsf{prer}^R_{\hat{J}^R}$. Namely, we sample $\mathsf{prer}^R_{\hat{J}^R} \xleftarrow{\$} \{0,1\}^{\lambda_{\mathsf{PRF}}}$. Note that we can repeat the argument we used from $\mathbf{G}_{3,b}$ to $\mathbf{G}_{4,b}$ and obtain

$$|\Pr[\mathbf{G}_{4,b} \Rightarrow 1] - \Pr[\mathbf{G}_{5,b} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{psrand}}_{\mathcal{B},\mathsf{PRF},d}(\lambda).$$

Game $\mathbf{G}_{6,b}$: In game $\mathbf{G}_{6,b}$, we change the way we compute $\mathsf{r}_{\hat{J}}$. Note that in $\mathbf{G}_{5,b}$ this was computed as $\mathsf{r}_{\hat{J}} := \mathsf{H}_x(\mathsf{prer}_{\hat{J}})$. Now, we sample it randomly as

$$\mathsf{r}_{\hat{J}} = (\alpha_{\hat{J}}, \beta_{\hat{J}}, \varphi_{\hat{J}}, \gamma_{\hat{J}}) \xleftarrow{\$} \mathcal{D} \times \mathbb{Z}_\gamma \times \mathcal{R}_{\mathsf{ck}} \times \{0,1\}^{\lambda_{\mathsf{PRF}}}.$$

Note that the adversary can only distinguish between games $\mathbf{G}_{5,b}$ and $\mathbf{G}_{6,b}$ if it queries $\mathsf{H}_x(\mathsf{prer}_{\hat{J}})$. However, $\mathcal{A}$ obtains no information about $\mathsf{prer}_{\hat{J}}$, which is sampled uniformly at random. By a union bound over all hash queries and $\{L, R\}$ we obtain

$$|\Pr[\mathbf{G}_{5,b} \Rightarrow 1] - \Pr[\mathbf{G}_{6,b} \Rightarrow 1]| \leq \frac{2Q_{\mathsf{H}_x}}{2^{\lambda_{\mathsf{PRF}}}}.$$

Game $\mathbf{G}_{7,b}$: Game $\mathbf{G}_{7,b}$ is as $\mathbf{G}_{6,b}$, except that it computes com_r in a different way. In detail, it samples $\eta \xleftarrow{\$} \{0,1\}^\lambda$ and computes the com_r as

$$\mathsf{com}_\mathsf{r} := \mathsf{H}_\mathsf{r}(\mathsf{H}_\mathsf{r}(\mathsf{r}_1), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_{\hat{J}-1}), \eta, \mathsf{H}_\mathsf{r}(\mathsf{r}_{\hat{J}+1}), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_N))$$

Later it returns η as part of its third message. Note that $\mathcal{A}$ can only see the difference between $\mathbf{G}_{6,b}$ and $\mathbf{G}_{7,b}$ if it queries $\mathsf{H}_\mathsf{r}(\mathsf{r}^X_{\hat{J}^X})$ for an $X \in \{L, R\}$. Note that $\mathcal{A}$ obtains no information about $\gamma_{\hat{J}}$ and $\gamma_{\hat{J}}$ is sampled uniformly at random. We can apply a union bound over all $Q_{\mathsf{H}_\mathsf{r}}$ random oracle queries and $X \in \{L, R\}$ and obtain

$$|\Pr[\mathbf{G}_{6,b} \Rightarrow 1] - \Pr[\mathbf{G}_{7,b} \Rightarrow 1]| \leq \frac{2Q_{\mathsf{H}_\mathsf{r}}}{2^{\lambda_{\mathsf{PRF}}}}.$$

Game $\mathbf{G}_{8,b}$: In game $\mathbf{G}_{8,b}$ we change the way the commitment $\mu_{\hat{J}}$ is generated. Recall that in $\mathbf{G}_{7,b}$, this is generated as

$$\mu_{\hat{J}} := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi_{\hat{J}}) = \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_0 + \varphi_{\hat{J}}).$$

Note that if the game does not stop, then especially $\hat{J} = J$ and $\varphi^* = \varphi_0 + \varphi_{\hat{J}}$. In game $\mathbf{G}_{8,b}$, we sample $\varphi^* \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}$ and set $\mu_{\hat{J}} := \mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi^*)$. We can argue that the view of $\mathcal{A}$ is unchanged as follows. Note that due to the previous changes, $\mathcal{A}$ gets no information about $\varphi_{\hat{J}}$. Thus, we have to consider the distribution of the value $\varphi^* = \varphi_0 + \varphi_{\hat{J}}$ conditioned on $k_{\hat{J}}, (\varphi_0 + \varphi_j)_{j \neq \hat{J}}$ and φ_0. This distribution is uniformly random as $\varphi_{\hat{J}}$ is uniformly random. Hence we have

$$\Pr[\mathbf{G}_{8,b} \Rightarrow 1] = \Pr[\mathbf{G}_{7,b} \Rightarrow 1].$$

Game $\mathbf{G}_{9,b}$: In game $\mathbf{G}_{9,b}$, we change the way μ_0 is generated. In particular, we sample a random message $\bar{\mathsf{m}}^L$ (resp. $\bar{\mathsf{m}}^R$) and set $\mu_0 := \mathsf{Com}(\mathsf{ck}, \bar{\mathsf{m}}; \varphi_0)$. Note that in $\mathbf{G}_{9,b}$ the value φ_0 is only needed to compute μ_0. Especially, it is not needed to compute the value φ^* due to the previous changes. It follows from the security of CMT that $\mathsf{Com}(\mathsf{ck}, \bar{\mathsf{m}}; \varphi_0)$ and $\mathsf{Com}(\mathsf{ck}, \mathsf{m}; \varphi_0)$ are identically distributed given ck. Therefore, the view of $\mathcal{A}$ is not affected by this change and we obtain

$$\Pr[\mathbf{G}_{9,b} \Rightarrow 1] = \Pr[\mathbf{G}_{8,b} \Rightarrow 1].$$

Note that in $\mathbf{G}_{9,b}$, the only part of the oracles $\mathrm{O}_0, \mathrm{O}_1$ that depends on bit b is the $\hat{J}^{th}$ session. Also, note that each session by itself corresponds to the user algorithm of the linear blind signature scheme from the OGQ linear function family, which is statistically blind. We show this in Lemma 3.5. Thus, we can reduce from the games in Lemma 3.5 to bound $\mathcal{A}$'s advantage in distinguishing between $\mathbf{G}_{9,0}$ and $\mathbf{G}_{9,1}$. To do so, we construct a reduction $\mathcal{B}'$ as follows:

- $\mathcal{B}'$ runs $\mathcal{A}$ and gets random coins ρ and messages $\mathsf{m}_0, \mathsf{m}_1$, i.e., $(\rho, \mathsf{m}_0, \mathsf{m}_1, St) \leftarrow \mathcal{A}(1^\lambda)$. $\mathcal{B}'$ partitions these coins into ρ_{ck} and ρ', where ρ_{ck} are the coins that determine ck and ρ' are the coins that determine the other parts of the key, i.e., par, pk' and $\mathsf{td}, \mathsf{sk}'$. It computes these values from the coins. Then, $\mathcal{B}'$ samples $\varphi_0^*, \varphi_1^* \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}$ and sets

  $$\mu_0^* := \mathsf{Com}(\mathsf{ck}, \mathsf{m}_0; \varphi_0^*), \quad \mu_1^* := \mathsf{Com}(\mathsf{ck}, \mathsf{m}_1; \varphi_1^*).$$

  It outputs ρ', μ_0^*, μ_1^* and its state to its challenger.

- $\mathcal{B}'$ is executed with access to oracles O_0' and O_1'. Also, $\mathcal{B}'$ has access to a random oracle H. $\mathcal{B}'$ simulates all random oracles except H'' honestly for $\mathcal{A}$, which involves appropriately forwarding queries from $\mathcal{A}$ to its challenger for oracle H. Oracle H'' is simulated as in game $\mathbf{G}_{9,b}, b \in \{0,1\}$, i.e., it is simulated honestly but the simulation is aborted whenever a collision occurs. It runs $\mathcal{A}$ on input St with access to random oracles and interactive oracles $\mathrm{O}_0, \mathrm{O}_1$, i.e., $St' \leftarrow \mathcal{A}^{\mathrm{O}_0, \mathrm{O}_1}(St)$. We describe the simulation of oracle O_0. Oracle O_1 is simulated analogously by using O_1' instead of O_0':

  - When $\mathcal{A}$ sends N, com_J, extract $\hat{J}$ from com_J as $\mathbf{G}_{9,b}, b \in \{0,1\}$ does. Sample keys $k \leftarrow \mathsf{PRF.Gen}(1^{\lambda_{\mathsf{PRF}}}, 1^{\log(N)}), k_{\hat{J}} \leftarrow \mathsf{PRF.Puncture}(k, \hat{J})$. Set $\mathsf{prer}_j := \mathsf{PRF.Eval}(k_{\hat{J}}, j)$ and set $\mathsf{r}_j := \mathsf{H}_x(\mathsf{prer}_j)$ for all $j \in [N] \setminus \{\hat{J}\}$. Sample $\varphi_0 \xleftarrow{\$} \mathcal{R}_{\mathsf{ck}}$ and a random message $\bar{\mathsf{m}}$ and set $\mu_0 := \mathsf{Com}(\mathsf{ck}, \bar{\mathsf{m}}; \varphi_0)$. Set $\mu_j := \mathsf{Translate}(\mathsf{ck}, \mu_0, \varphi_j)$ for $j \in [N] \setminus \{\hat{J}\}$.. Sample $\eta \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathsf{com}_\mathsf{r} := \mathsf{H}_\mathsf{r}(\mathsf{H}_\mathsf{r}(\mathsf{r}_1), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_{\hat{J}-1}), \eta, \mathsf{H}_\mathsf{r}(\mathsf{r}_{\hat{J}+1}), \ldots, \mathsf{H}_\mathsf{r}(\mathsf{r}_N))$. Send μ_0 and com_r to $\mathcal{A}$.
  - Obtain seed from adversary and compute all c_j for $j \in [N] \setminus \{\hat{J}\}$ as in the scheme using the values μ_j. For session $\hat{J}$, compute $R_{\hat{J}} := \mathsf{H}'(\mathsf{seed}, \hat{J})$ and send $R_{\hat{J}}$ to oracle O_0'. Obtain a value $c_{\hat{J}}$ and compute $\mathsf{com}_c := \mathsf{H}_c(c_1, \ldots, c_N)$. Send com_c to $\mathcal{A}$.
  - Obtain J, salt from $\mathcal{A}$. If $\mathsf{com}_J \neq \mathsf{H}''(J, \mathsf{salt})$ abort the execution of this oracle. Otherwise it must holds that $\hat{J} = J$. Continue by sending $k_{\hat{J}}, c_{\hat{J}}$ and η to $\mathcal{A}$.
  - Obtain s_J from $\mathcal{A}$ and forward it to oracle O_0'.

- $\mathcal{B}'$ obtains signatures $\sigma'_0 = (c'_0, s'_0)$ and $\sigma'_1 = (c'_1, s'_1)$ from its challenger and sets

$$\sigma_0 := (c'_0, s'_0, \varphi^*_0), \quad \sigma_1 := (c'_1, s'_1, \varphi^*_1).$$

  It runs $b' \leftarrow \mathcal{A}(St', \sigma_0, \sigma_1)$ and returns b' to the challenger.

It is easy to see that if $\mathcal{B}'$ runs in game $\mathbf{G}_0$ from Lemma 3.5, then it perfectly simulates game $\mathbf{G}_{9,0}$ for $\mathcal{A}$, and if it runs in game $\mathbf{G}_1$ from Lemma 3.5 it perfectly simulates game $\mathbf{G}_{9,1}$ for $\mathcal{A}$. Also, $\mathcal{B}'$'s output is the output of $\mathcal{A}$ and $\mathcal{B}'$ makes as many random oracle queries as $\mathcal{A}$ does (with respect to random oracle H). We know by Lemma 3.5 that $\mathcal{B}'$ has advantage in distinguishing the games $\mathbf{G}_0$ and $\mathbf{G}_1$ at most $4Q_{\mathsf{H}}/|\mathbb{Z}^*_n|$. Hence we have

$$|\Pr[\mathbf{G}_{9,0} \Rightarrow 1] - \Pr[\mathbf{G}_{9,1} \Rightarrow 1]| \leq \frac{4Q_{\mathsf{H}}}{|\mathbb{Z}^*_n|}.$$

The statement follows from an easy calculation. □

Theorem 3.5. *Let* PRF *be a puncturable pseudorandom function and* CMT *be a randomness homomorphic commitment scheme. Further, let* $\mathsf{H}' \colon \{0,1\}^* \to \mathbb{Z}^*_n, \mathsf{H}'' \colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\mathsf{r}, \mathsf{H}_c \colon \{0,1\}^* \to \{0,1\}^\lambda$ *be random oracles. Then,* $\mathsf{BS}_{\mathsf{RSA}}$ *satisfies one-more unforgeability, assuming that* CCBS[CMT] *(cf. Section 3.5.2) does.*

Specifically, for any $L = \mathsf{poly}(\lambda)$*, any PPT algorithm* $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_c}, Q_{\mathsf{H}'}, Q_{\mathsf{H}''}, Q_{\mathsf{H}}$ *queries to oracles* $\mathsf{H}_\mathsf{r}, \mathsf{H}_c, \mathsf{H}', \mathsf{H}'', \mathsf{H}$*, respectively, and starts at most* p *interactions with his signer oracle, there exists a PPT algorithm* $\mathcal{B}$ *that makes at most* Q_{H} *queries to* H*, starts at most* p *interactions with his signer oracle, makes at most* p^2 *queries to his oracle* $\hat{\mathsf{H}}$*, such that* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$*, and*

$$\mathsf{Adv}^{L\text{-OMUF}}_{\mathcal{A},\mathsf{BS}_{\mathsf{RSA}}}(\lambda) \leq \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_c}}{2^\lambda} + \frac{pQ_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{pQ_{\mathsf{H}_c}}{2^\lambda} + \frac{pQ_{\mathsf{H}'}}{2^\lambda} + \frac{pQ_{\mathsf{H}''}}{2^\lambda} + \mathsf{Adv}^{L\text{-OMUF}}_{\mathcal{B},\mathsf{CCBS}[\mathsf{CMT}]}(\lambda).$$

Proof. Set $\mathsf{BS} := \mathsf{BS}_{\mathsf{RSA}}$. Let $\mathcal{A}$ be an adversary against the OMUF security of BS. We prove the statement via a sequence of games. Some parts of the proof are taken verbatim from the proof of Theorem 3.2. Fix an arbitrary polynomial ℓ.

Game $\mathbf{G}_0$: We start with game $\mathbf{G}_0 := \ell\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}$, which is the one-more unforgeability game. We briefly recall this game. First, a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is sampled and $\mathcal{A}$ is run with concurrent access to an interactive oracle O simulating $\mathsf{S}(\mathsf{sk})$. We denote the number of completed interactions between $\mathcal{A}$ and O after $\mathcal{A}$'s execution by ℓ. As we consider the random oracle model, $\mathcal{A}$ also gets access to random oracles, which are provided by the game in the standard lazy manner. When $\mathcal{A}$ finishes its execution, it outputs tuples $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$ and wins, if all m_i are distinct, $k > \ell$ and all signatures σ_i verify with respect to pk and m_i.

Game $\mathbf{G}_1$: This game is as $\mathbf{G}_0$, but we rule out collisions for oracles $\mathsf{H}_t, t \in \{\mathsf{r}, c\}$. To be more precise, we change the simulation of oracles $\mathsf{H}_t, t \in \{\mathsf{r}, c\}$ in the following way. If $\mathcal{A}$ queries $\mathsf{H}_t(x)$ and this value is not yet defined, the game samples an image $y \xleftarrow{\$} \{0,1\}^\lambda$. However, if there exists an $x' \neq x$ with $\mathsf{H}_t(x') = y$, the game returns $\perp$. Otherwise it behaves as before. Note that $\mathcal{A}$ can only distinguish between $\mathbf{G}_0$ and $\mathbf{G}_1$ if such a collision happens, i.e., H_t returns $\perp$. We can apply a union bound over all $Q^2_{\mathsf{H}_t}$ pairs of random oracle queries and obtain

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_c}}{2^\lambda}.$$

In particular, the previous change implies that at each point of the execution of the game and for each image $y \in \{0,1\}^\lambda$, there is at most one preimage $\mathsf{H}_t^{-1}(y)$ under H_t. Thus, from a given image

$y \in \{0,1\}^\lambda$, the game can extract at most one preimage $x \in \{0,1\}^*$ such that $\mathsf{H}_t(x) = y$. We will use this in the following games.

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, we change the way the signing oracle is executed. Whenever $\mathcal{A}$ sends $\mathsf{com}_\mathsf{r}, \mu_0$ as its first message, the game tries to extract the messages from this commitment. To do so, the game goes through all the random oracle queries to H_r and tries to find a preimage of com_r. Then, it parses this bitstring as N hashes $h_1, \ldots, h_N$ and tries to find preimages of these in the same way. As a result of this extraction, the game will end up with values $\bar{\mathsf{r}}_1, \ldots, \bar{\mathsf{r}}_N$, where we write $\bar{\mathsf{r}}_j = \perp$ if the game was not able to extract the j^{th} value. If there is a session $j \in [N]$ such that $\bar{\mathsf{r}}_j = \perp$, i.e., the game could not extract the randomness for that session, and later in that execution $J \neq j$ but algorithm Check outputs 1, the game aborts. Denote this event by bad_1. The probability of bad_1 is an upper bound on the distinguishing advantage of $\mathcal{A}$ between $\mathbf{G}_1$ and $\mathbf{G}_2$. For each fixed interaction, we can bound the probability of this event (with respect to that interaction) by $Q_{\mathsf{H}_\mathsf{r}}/2^\lambda$. To see this, consider the case where the game could not extract the values $h_1, \ldots, h_N$. Then, once com_r is fixed, the probability that one of the hash queries of $\mathcal{A}$ evaluates to com_r is at most $1/2^\lambda$. Similarly, in the case where the game could extract h_j but could not extract a value $\bar{\mathsf{r}}_j$ from h_j for $J \neq j$, the probability that one of the hash queries of $\mathcal{A}$ evaluates to the fixed h_j is at most $1/2^\lambda$. By a union bound over all interactions we obtain

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \Pr[\mathsf{bad}_1] \leq \frac{pQ_{\mathsf{H}_\mathsf{r}}}{2^\lambda}.$$

Game $\mathbf{G}_3$: Again, we change the signing oracle by introducing an additional abort. Namely, whenever the adversary sends the commitment com_c as its second message, the game extracts values $\bar{c}_1, \ldots, \bar{c}_N$ such that $\mathsf{H}_c(\bar{c}_1, \ldots, \bar{c}_N) = \mathsf{com}_c$ by going through all random oracle queries of $\mathcal{A}$. If the game is not able to extract, but later algorithm Check outputs 1, the game aborts and we say that the event bad_2 occurs. Note that algorithm Check internally checks if

$$\mathsf{com}_c \neq \mathsf{H}_c(c_1, \ldots, c_N).$$

Thus, the probability of bad_2 in a fixed interaction and hence the distinguishing advantage of $\mathcal{A}$ between $\mathbf{G}_2$ and $\mathbf{G}_3$ is bounded by $Q_{\mathsf{H}_c}/2^\lambda$, using a similar argument as in the previous game. We obtain

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \Pr[\mathsf{bad}_2] \leq \frac{pQ_{\mathsf{H}_c}}{2^\lambda}.$$

Game $\mathbf{G}_4$: This game aborts whenever the following bad event occurs. The event is defined as follows: The game samples seed after $\mathcal{A}$ sends its first message of an interaction with the signer oracle and at this point there exists an index $j \in [N]$ such that $\mathsf{H}'(\mathsf{seed}, j)$ is already defined. As seed is sampled uniformly at random from $\{0,1\}^\lambda$ and hidden from $\mathcal{A}$ until the point of the potential abort, a union bound over all hash queries and interactions shows that

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \frac{pQ_{\mathsf{H}'}}{2^\lambda}.$$

Game $\mathbf{G}_5$: In $\mathbf{G}_5$, the signer oracle sends a random com_J in the beginning of each interaction. Later, before it has to send J, salt, it samples $J \xleftarrow{\$} [N]$ and $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ and aborts if $\mathsf{H}''(J, \mathsf{salt})$ is already defined. If it is not yet defined, it defines it as $\mathsf{H}''(J, \mathsf{salt}) := \mathsf{com}_J$. The adversary $\mathcal{A}$ can only distinguish between $\mathbf{G}_4$ and $\mathbf{G}_5$ if $\mathsf{H}''(J, \mathsf{salt})$ is already defined. By a union bound over all $Q_{\mathsf{H}''}$ hash queries and p interactions we obtain

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{pQ_{\mathsf{H}''}}{2^\lambda}.$$

Let us summarize what we have so far. We changed the game step by step and ruled out the following types of bad events:

1. The adversary sends some commitment for which the game can not extract some of the committed values, but later the adversary can open it successfully.
2. The game samples a random seed such that the random oracle values of interest are already defined for that seed.

In particular, from the first property we can derive that whenever the game does not abort, it could successfully extract values all of the values $\bar{c}_1, \ldots, \bar{c}_N$. Additionally, we know by the collision freeness of H_c that we must have $c_j = \bar{c}_j$ for all $j \in [N]$. A similar statement holds for the $\bar{\mathsf{r}}_j$. Here, it can only be the case that the game can not extract a single $\bar{\mathsf{r}}_j$ but later $J = j$. On the other hand, the second property tells us that a potential reduction simulating $\mathbf{G}_5$ can program the random oracle before sending the seed or the cut-and-choose index J to the adversary. We will use these properties to construct a (tight) reduction $\mathcal{B}$ that breaks the one-more unforgeability of $\mathsf{CCBS}[\mathsf{CMT}]$ whenever $\mathbf{G}_5$ outputs 1. Reduction $\mathcal{B}$ works as follows:

- $\mathcal{B}$ gets as input $\mathsf{pk} = (\mathsf{par}, \mathsf{pk}', \mathsf{ck})$ and oracle access to a signer oracle $\hat{\mathsf{O}}$ and random oracles $\mathsf{H}, \hat{\mathsf{H}}$ for blind signature scheme $\mathsf{CCBS}[\mathsf{CMT}]$. It runs $\mathcal{A}$ with input pk and oracle access to random oracles $\mathsf{H}, \mathsf{H}', \mathsf{H}'', \mathsf{H}_\mathsf{r}$ and H_c and a signer oracle O. The oracles $\mathsf{H}', \mathsf{H}''$ are simulated honestly by $\mathcal{B}$ and oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_c$ are simulated exactly as in game $\mathbf{G}_5$.
- When adversary $\mathcal{A}$ queries oracle O to start an interaction, the reduction $\mathcal{B}$ behaves as follows:
  - It starts an interaction with oracle $\hat{\mathsf{O}}$ and obtains a parameter N as the first message. It forwards N, com_J to $\mathcal{A}$, where $\mathsf{com}_J \xleftarrow{\$} \{0,1\}^\lambda$.
  - When $\mathcal{A}$ sends its first message $\mathsf{com}_\mathsf{r}, \mu_0$, $\mathcal{B}$ extracts $(\bar{\mathsf{r}}_1, \ldots, \bar{\mathsf{r}}_N)$ as in game $\mathbf{G}_5$ (cf. $\mathbf{G}_2$). For each such $j \in [N]$ for which $\bar{\mathsf{r}}_j$ is defined, it parses $\bar{\mathsf{r}}_j = (\bar{\alpha}_j, \bar{\beta}_j, \bar{\varphi}_j, \bar{\gamma}_j)$ and sets $\bar{\mu}_j := \mathsf{Translate}(\mathsf{ck}, \mu_0, \bar{\varphi}_j)$. Then it defines $\mathsf{com}_j := \hat{\mathsf{H}}(\bar{\alpha}_j, \bar{\beta}_j, \bar{\mu}_j, \bar{\gamma}_j)$. For the remaining j, it samples $\mathsf{com}_j \xleftarrow{\$} \{0,1\}^\lambda$. Finally, it sends $\mathsf{com}_1, \ldots, \mathsf{com}_N$ to its oracle $\hat{\mathsf{O}}$.
  - The oracle $\hat{\mathsf{O}}$ returns $R_1, \ldots, R_N$. Then, $\mathcal{B}$ samples $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. It aborts, if there exists an index $j \in [N]$ such that $\mathsf{H}'(\mathsf{seed}, j)$ is already defined. Otherwise, it programs $\mathsf{H}'(\mathsf{seed}_R, j) := R_j$ for all $j \in [N]$ and sends seed to $\mathcal{A}$.
  - When $\mathcal{A}$ sends its second message com_c, the game extracts values $\bar{c}_i$ from com_c. If this extraction is successful, i.e., $\bar{c}_j$ is defined, it sets $c'_j := \bar{c}_j$. Otherwise, it sets $c'_j \xleftarrow{\$} \mathcal{S}$. It sends $c'_1, \ldots, c'_N$ to $\hat{\mathsf{O}}$.
  - The oracle $\hat{\mathsf{O}}$ returns an index $J \in [N]$, whereupon reduction $\mathcal{B}$ samples $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ and aborts if $\mathsf{H}''(J, \mathsf{salt})$ is already defined. Otherwise it sets $\mathsf{H}''(J, \mathsf{salt}) := \mathsf{com}_J$ and sends J, salt to $\mathcal{A}$.
  - When adversary sends its third message k_J, c_J, η, algorithm $\mathcal{B}$ runs algorithm Check. If this algorithm returns 0, $\mathcal{B}$ aborts this interaction. If it outputs 1 it aborts the entire execution if one of the following two conditions hold
    - There is some index $j \in [N]$ such that $c_j = \bot$.
    - There is some index $j \in [N]$ such that $j \neq J$ and $\bar{\mathsf{r}}_j = \bot$.

    Otherwise, $\mathcal{B}$ sends $\{(\bar{\alpha}_j, \bar{\beta}_j, \bar{\mu}_j, \bar{\gamma}_j)\}_{j \neq J}$ to $\hat{\mathsf{O}}$. Note that these values are defined by the second condition that has been checked before.
  - The oracle $\hat{\mathsf{O}}$ returns s_J and $\mathcal{B}$ forwards it to $\mathcal{A}$.
- When $\mathcal{A}$ outputs $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$, $\mathcal{B}$ outputs $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$.

It is easy to see that the values $R_1, \ldots, R_N$ are distributed uniformly over $\mathbb{Z}_n^*$. This property of the OGQ function is called smoothness in [HKL19]. Therefore, the programming of the random oracle H' is done correctly. Further, we claim that whenever $\mathcal{B}$ does not abort during the interaction, the signing

oracle $\hat{\mathsf{O}}$ will also not abort. From this claim it follows that the simulation provided by $\mathcal{B}$ is perfect. To see that the claim is true, note that $\hat{\mathsf{O}}$ could abort the signing interaction for two reasons: First, it may abort as there exists some $j \in [N]$ such that $j \neq J$ and $\mathsf{com}_j \neq \hat{\mathsf{H}}(\bar{\alpha}_j, \bar{\beta}_j, \bar{\mu}_j, \bar{\gamma}_j)$. However, this can not happen due to the way $\mathcal{B}$ defines com_j. The second reason for an abort is that there exists a $j \in [N]$ such that $j \neq J$ and $c'_j \neq \mathsf{H}(\bar{\mu}_j, R_j \cdot \mathsf{F}(\bar{\alpha}_j) \cdot \mathsf{pk}'^{\bar{\beta}_j}) + \bar{\beta}_j$. However, as we already noticed above, if $\mathbf{G}_6$ does not abort, then we have $c'_j = c_j, \bar{\mu}_j = \mu_j, \bar{\alpha}_j = \alpha_j$ and $\bar{\beta}_j = \beta_j$ and thus the $\mathcal{B}$ itself would have aborted as Check returns 0. Finally, it is clear that $\mathcal{B}$ wins the one-more unforgeability game whenever $\mathbf{G}_6$ outputs 1, as $\mathcal{B}$ outputs $\mathcal{A}$'s output and completes as many interactions with oracle $\hat{\mathsf{O}}$ as $\mathcal{A}$ completes with O. The statement follows by an easy calculation. □

3.5.4 Concrete Parameters and Efficiency

To derive concrete parameters for our scheme $\mathsf{BS}_{\mathsf{RSA}}$ based on the RSA assumption in a theoretically sound way, we recall the concrete security bounds from Theorems 3.3 and 3.5. Let ϵ, t denote the advantage and running time of an adversary against the one-more unforgeability of $\mathsf{BS}_{\mathsf{RSA}}$ initiating at most p interactions with the signing oracle and querying the random oracle at most Q_{H} many times. Then there is an adversary against the one-more unforgeability $\mathsf{CCBS}[\mathsf{CMT}]$ with advantage ϵ_{CCBS} and running time t. Also, there are three algorithms solving two instances of the RSA problem with probability $\epsilon_{\mathsf{RSA}}, \epsilon_{\mathsf{RSA}'}, \epsilon_{\mathsf{RSA}''}$ and running time $2t, t, t$, respectively. Here, the third adversary against RSA comes from the binding property of the commitment scheme we use.

Concretely, by combining the concrete bounds given in Theorems 3.3 and 3.5 we obtain that

$$\epsilon \leq 2\sqrt[3]{Q_{\mathsf{H}}^2 \ell^3 \epsilon_{\mathsf{RSA}}} + \frac{(Q_{\mathsf{H}}(p-\ell))^{\ell+1}}{\gamma} + 2\epsilon_{\mathsf{RSA}''} + 2p\epsilon_{\mathsf{RSA}'} + 2T_1 + T_2,$$

where T_1, T_2 are statistically negligible terms and $\ell = 3\ln(p+1) + \ln(2/\epsilon_{\mathsf{CCBS}})$. To simplify further, we assume κ bit of security for the instance related to ϵ_{RSA} and $\epsilon_{\mathsf{RSA}'}$ and κ'' bit of security for the instance related to $\epsilon_{\mathsf{RSA}''}$. By definition, this means that

$$\epsilon_{\mathsf{RSA}} < 2 \cdot t \cdot 2^{-\kappa},\ \epsilon_{\mathsf{RSA}'} < t \cdot 2^{-\kappa},\ \epsilon_{\mathsf{RSA}''} < t \cdot 2^{-\kappa''}.$$

Next, we use

$$\ell = 3\ln(p+1) + \ln\left(\frac{2}{\epsilon_{\mathsf{CCBS}}}\right) \leq 3\ln(p+1) + \ln\left(\frac{2}{\epsilon - T_2}\right) =: \ell_\epsilon.$$

Plugging in, we get

$$\epsilon \leq 2\sqrt[3]{Q_{\mathsf{H}}^2 \ell_\epsilon^3 \cdot 2 \cdot t \cdot 2^{-\kappa}} + \frac{(Q_{\mathsf{H}} p)^{\ell_\epsilon+1}}{\gamma} + 2t \cdot 2^{-\kappa''} + 2pt \cdot 2^{-\kappa} + 2T_1 + T_2,$$

which must hold for any adversary with running time t and advantage ϵ and any γ we choose. Note that we can set the bitlength of the prime γ independently of the RSA modulus length.

To get k bit of security for $\mathsf{BS}_{\mathsf{RSA}}$, we consider any fixed choice of ϵ, t such that $t/\epsilon = 2^k$ and increase κ, κ'' until the above inequality leads to a contradiction. Then, we choose the maximum values for κ, κ''. We note that we have to take this two-step approach and iterate over all combinations of ϵ, t, as ℓ_ϵ depends on ϵ which leads to a non-linear inequality. Also, we note that we can set κ'' to be much less than κ as the relation between k and κ'' is tight. Once the appropriate security levels κ and κ'' are found, we determine the modulus lengths $\mathsf{len}, \mathsf{len}''$ following an estimation for the sub-exponential complexity of the general number field sieve algorithm [CP06], which is similar to [HKL19]. Using the modulus length and the bitlength of γ, we can compute the sizes of signatures and keys.

Next, we consider blindness. For simplicity, assume that $N^L = N^R =: N$. Also, we can make the assumption[9] that $|\mathbb{Z}_n^*| \geq 2^\lambda$. If we want to achieve blindness with k bits of security, we have to make sure that the blindness advantage is at most $2^{-k} \cdot t$. As for our CDH-based scheme, we instantiate PRF using the GGM construction (cf. Section 3.3). Using Theorem 3.4, the blindness advantage can be upper bounded by

$$\frac{(2\log(N)-1)Q_{\mathsf{H}_{\mathsf{PRF}}}}{2^{\lambda_{\mathsf{PRF}}-2}} + \frac{Q_{\mathsf{H}''}^2}{2^{\lambda-1}} + \frac{Q_{\mathsf{H}''}}{2^{\lambda-2}} + \frac{Q_{\mathsf{H}_x}}{2^{\lambda_{\mathsf{PRF}}-2}} + \frac{Q_{\mathsf{H}_r}}{2^{\lambda_{\mathsf{PRF}}-2}} + \frac{Q_{\mathsf{H}}}{2^{\lambda-2}}.$$

Thus, we only have to choose λ_{PRF} large enough. We implemented the approach in a simple Python script[10]. Example instantiations of our parameters can be found in Table 3.1.

3.6 Intermezzo: From Semi-Honest to Malicious Blindness

In this short intermezzo, we show how to transform any blind signature scheme with semi-honest signer blindness into a scheme that satisfies malicious signer blindness. In summary, we show the following lemma.

Lemma 3.6 (Informal)**.** *Let* BS *be a blind signature scheme that satisfies semi-honest signer blindness. Then, using a non-interactive zero-knowledge proof-of-knowledge,* BS *can be transformed into a blind signature scheme* BS′ *that satisfies malicious signer blindness. Furthermore, the schemes* BS *and* BS′ *are identical except for public keys* pk*. Finally, for any* $\ell\colon \mathbb{N} \to \mathbb{N}$*, it holds that if* BS *satisfies* ℓ-OMUF*, then* BS′ *satisfies* ℓ-OMUF *and the security proofs of this transformation are tight.*

Construction. Let $\mathsf{BS} = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ be a blind signature scheme. Consider the relation of public keys and random coins

$$\mathcal{R} = \left\{(\mathsf{pk}, \rho) \;\middle|\; \exists\, \mathsf{sk} : (\mathsf{pk}, \mathsf{sk}) = \mathsf{Gen}(1^\lambda; \rho)\right\},$$

where pk is the statement and ρ is the witness. Further, let $\mathsf{PS} = (\mathsf{PProve}, \mathsf{PVer})$ be a NIZKPOK for relation $\mathcal{R}$ using random oracle H. We transform BS into a new blind signature scheme $\mathsf{BS}' = (\mathsf{Gen}', \mathsf{S}, \mathsf{U}', \mathsf{Ver})$, where algorithms S and Ver stay the same. Algorithm Gen′ is as follows:

1. Sample random coins ρ for algorithm Gen.
2. Generate $(\mathsf{pk}_0, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda; \rho)$.
3. Compute a proof $\pi \leftarrow \mathsf{PProve}^{\mathsf{H}}(\mathsf{pk}_0, \rho)$ using pk_0 as the statement and ρ as the witness.
4. Return the public key $\mathsf{pk} := (\mathsf{pk}_0, \pi)$ and the secret key sk.

Further, algorithm U′ first checks the correctness of pk. That is, U′ parses $\mathsf{pk} = (\mathsf{pk}_0, \pi)$ and runs $b := \mathsf{PVer}^{\mathsf{H}}(\mathsf{pk}_0, \pi)$. Then, if $b = 0$, it aborts the interaction. If $b = 1$, it behaves as U does.

Analysis. Below, we show that the scheme BS′ is malicious signer blind, assuming BS is semi-honest signer blind and PS has the proof-of-knowledge property. We also show that BS′ is one-more unforgeable, if BS is one-more unforgeable and PS is zero-knowledge.

Lemma 3.7. *Let* BS *be a blind signature scheme and* PS *be a NIZKPOK for the relation* $\mathcal{R}$ *using random oracle* H*. Then,* BS′ *satisfies malicious signer blindness assuming that* BS *satisfies semi-honest signer blindness.*

[9] This is without loss of generality, as we have to choose a security level larger than λ for the underlying RSA levels.

[10] The Python script can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.

Concretely, let Ext *be the extractor algorithm given by the proof-of-knowledge property of* PS*. Then, for any algorithm* $\mathcal{A}$ *against the malicious signer blindness of* BS'*, there exist an algorithm* $\mathcal{B}$ *against the semi-honest signer blindness and an algorithm* $\mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}) + \mathbf{T}(\mathsf{Ext})$, $\mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$*, and*

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}'}(\lambda) \leq \mathsf{Adv}^{\mathsf{blind}}_{\mathcal{B},\mathsf{BS}}(\lambda) + \mathsf{Adv}^{\mathsf{pok}}_{\mathcal{B}',\mathsf{PS},\mathsf{Ext}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary against malicious signer blindness of BS'. First, we can assume that $\mathcal{A}$ outputs a key $\mathsf{pk} = (\mathsf{pk}_0, \pi)$ such that $\mathsf{PVer}^{\mathsf{H}}(\mathsf{pk}_0, \pi) = 1$. This is because otherwise, both user oracles of scheme BS' abort and leak no information about the message that is used.

With this assumption in mind, we build a reduction $\mathcal{B}$ against semi-honest signer blindness of BS. The reduction uses $\mathcal{A}$ as a subroutine. It also makes use of the extractor Ext that exists due to the proof-of-knowledge property of PS. Reduction $\mathcal{B}$ is as follows:

- $\mathcal{B}$ simulates the random oracle H for $\mathcal{A}$, keeping track of the list $\mathcal{Q}$ of random oracle queries and answers.
- $\mathcal{B}$ obtains a key pk and messages $\mathsf{m}_0, \mathsf{m}_1$ from $\mathcal{A}$. It parses $\mathsf{pk} = (\mathsf{pk}_0, \pi)$ and runs $\rho \leftarrow \mathsf{Ext}(x, \pi, \mathcal{Q})$. If $(\mathsf{pk}_0, \rho) \notin \mathcal{R}$, $\mathcal{B}$ sets $\mathsf{bad} = 1$ and simulates random oracles $\mathrm{O}'_0, \mathrm{O}'_1$ to adversary $\mathcal{A}$ by aborting each interaction. Later it returns $\sigma_0 = \bot, \sigma_1 = \bot$ to $\mathcal{A}$. Otherwise, if $(\mathsf{pk}_0, \rho) \in \mathcal{R}$, $\mathcal{B}$ outputs the random coins ρ and the messages $\mathsf{m}_0, \mathsf{m}_1$ to its blindness game. Note that this implies that its blindness game will use the public key pk_0.
- $\mathcal{B}$ gets access to oracles $\mathrm{O}_0, \mathrm{O}_1$. It provides oracles $\mathrm{O}'_0, \mathrm{O}'_1$ to adversary $\mathcal{A}$. By our assumption, we have $\mathsf{PVer}^{\mathsf{H}}(\mathsf{pk}_0, \pi) = 1$. Reduction $\mathcal{B}$ simulates O'_0 as O'_0 and O'_1 as O'_1.
- It runs $\mathcal{A}$ with one-time access to oracles $\mathrm{O}'_0, \mathrm{O}'_1$. Then, it obtains signatures σ_0, σ_1 from its blindness game and forwards them to $\mathcal{A}$.
- When $\mathcal{A}$ outputs a bit b', $\mathcal{B}$ forwards this bit to its blindness game.

It is clear that the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$ and the running time of Ext. Further, assuming that bad is never set to 1, $\mathcal{B}$ perfectly simulates the malicious signer blindness game $\mathbf{BLIND}_{b,\mathsf{BS}'}$ for $\mathcal{A}$ if it runs in semi-honest signer blindness game $\mathbf{BLIND}_{b,\mathsf{BS}}$. Also, note that the probability of $\mathsf{bad} = 1$ is bounded by the proof-of-knowledge property of PS using another reduction $\mathcal{B}'$. Further, the probability of $\mathsf{bad} = 1$ is independent of the bit b. To conclude the proof, we define the event $\mathsf{W}_b := \left(\mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}'}(\lambda) \Rightarrow 1\right)$. We obtain

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \quad & |\Pr[\mathsf{W}_0] - \Pr[\mathsf{W}_1]| \\
\leq \quad & |\Pr[\mathsf{W}_0 \mid \mathsf{bad} = 0] \cdot \Pr[\mathsf{bad} = 0] + \Pr[\mathsf{W}_0 \mid \mathsf{bad} = 1] \cdot \Pr[\mathsf{bad} = 1] \\
& - \Pr[\mathsf{W}_1 \mid \mathsf{bad} = 0] \cdot \Pr[\mathsf{bad} = 0] + \Pr[\mathsf{W}_1 \mid \mathsf{bad} = 1] \cdot \Pr[\mathsf{bad} = 1]| \\
\leq \quad & |\Pr[\mathsf{W}_0 \mid \mathsf{bad} = 0] - \Pr[\mathsf{W}_1 \mid \mathsf{bad} = 0]| \cdot \Pr[\mathsf{bad} = 0] \\
& + |\Pr[\mathsf{W}_0 \mid \mathsf{bad} = 1] - \Pr[\mathsf{W}_1 \mid \mathsf{bad} = 1]| \cdot \Pr[\mathsf{bad} = 1] \\
\leq \quad & |\Pr[\mathsf{W}_0 \mid \mathsf{bad} = 0] - \Pr[\mathsf{W}_1 \mid \mathsf{bad} = 0]| + \Pr[\mathsf{bad} = 1] \\
\leq \quad & \mathsf{Adv}^{\mathsf{blind}}_{\mathcal{B},\mathsf{BS}}(\lambda) + \mathsf{Adv}^{\mathsf{pok}}_{\mathcal{B}',\mathsf{PS},\mathsf{Ext}}(\lambda).
\end{aligned}$$

□

Lemma 3.8. *Let* BS *be a blind signature scheme,* PS *be a NIZKPOK for the relation* $\mathcal{R}$ *using random oracle* H*, and* $\ell \colon \mathbb{N} \to \mathbb{N}$ *be a function. Then,* BS' *satisfies* ℓ*-one-more unforgeability assuming that* BS *satisfies* ℓ*-one-more unforgeability.*

Concretely, let Sim *be the simulator algorithm given by the zero-knowledge property of* PS*. Then, for any PPT algorithm* $\mathcal{A}$ *against the* ℓ*-one-more unforgeability of* BS'*, there exist PPT algorithm* $\mathcal{B}$ *and* $\mathcal{B}'$ *such that* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}) + \mathbf{T}(\mathsf{Sim})$, $\mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$*, and*

$$\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}'}(\lambda) \leq \mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{B},\mathsf{BS}}(\lambda) + \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B}',\mathsf{PS},\mathsf{Sim}}(\lambda).$$

Proof. Let $\mathcal{A}$ be an adversary against the ℓ-OMUF security of BS'. We show the statement via a reduction from the ℓ-OMUF security of BS.
Game $\mathbf{G}_0$: We start with game $\mathbf{G}_0 := \ell\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}'}$, which is the one-more unforgeability game. First, a key pair $(\mathsf{pk}, \mathsf{sk})$ is sampled by the game. Recall that pk is of the form $\mathsf{pk} = (\mathsf{pk}_0, \pi)$, where $\pi \leftarrow \mathsf{PProve}^{\mathsf{H}}(\mathsf{pk}_0, \rho)$ and ρ are the random coins used to generate pk_0. Then, $\mathcal{A}$ is executed with input pk and oracle access to random oracle H and a signer oracle O'. In the end, $\mathcal{A}$ outputs pairs of signatures and messages and the game outputs 1 if these are all valid, the messages are distinct and there are more such pairs than the number of completed interactions with oracle O'. By definition, we have

$$\mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{A},\mathsf{BS}'}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_1$: We change the way the proof π contained in pk is generated. Namely, we use the simulator Sim that exists by the zero-knowledge property of PS to generate π. Note that this simulator may program the random oracle H. Clearly, we can bound the difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ by the advantage of a reduction $\mathcal{B}'$ against the zero-knowledge property of PS. Thus, we have

$$\left|\Pr\left[\mathbf{G}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1 \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{B}',\mathsf{PS},\mathsf{Sim}}(\lambda).$$

We will now bound the probability that $\mathbf{G}_1$ outputs 1 by a reduction $\mathcal{B}$ from the ℓ-OMUF security of BS. We denote the advantage of $\mathcal{B}$ by ϵ_{BS}. The reduction is as follows:

- $\mathcal{B}$ gets as input a public key pk_0 for scheme BS. Also, $\mathcal{B}$ gets oracle access to a signer oracle O for scheme BS. It generates a proof π for the statement pk_0 using the zero-knowledge simulator Sim and defines $\mathsf{pk} := (\mathsf{pk}_0, \pi)$.
- $\mathcal{B}$ runs algorithm $\mathcal{A}$ on input pk by providing random oracle H and a signer oracle O', which is simulated using O.
- $\mathcal{B}$ forwards the outputs of $\mathcal{A}$ to its own game.

It is easy to see that the reduction perfectly simulates $\mathbf{G}_1$ for $\mathcal{A}$. Also, as verification for BS is the same as for BS', any valid forgery of $\mathcal{A}$ leads to a valid forgery of $\mathcal{B}$. Thus, we have

$$\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] \leq \mathsf{Adv}^{\ell\text{-}\mathsf{OMUF}}_{\mathcal{B},\mathsf{BS}}(\lambda).$$

□

Applicability. Let us discuss how to apply the transformation that we presented in this section to our concrete scheme from RSA. We can use a generic zero-knowledge proof-of-knowledge for **NP** to apply the transformation to any blind signature scheme that satisfies semi-honest signer blindness. While this is an inefficient solution in general, we argue that it is acceptable here. First, such proofs are allowed to use random oracles in our setting. Second, the proof only needs to be generated once, and verified once by each user. Therefore, this will only induce a one-time overhead, which is independent of the complexity of the actual signing protocol.

Taking into account the concrete structure of the schemes we construct, more efficient solutions are possible. For example, the relation between a public and a secret key (which is part of the random coins used for key generation) in our scheme is given by a linear function. Therefore, simple Schnorr-style Fiat-Shamir [FS87] proofs can be used. In an RSA-based setting, one also needs to prove knowledge of the coins needed for the generation of parameters. Now, consider our specific RSA-based scheme from the OGQ linear function family, as presented in Section 3.5. Here, the random coins used for key generation are given by $p, q, a, \gamma, \mathsf{sk}'$ and the random coins used to generate ck. First, one can show that n is the product of two primes p, q using techniques from [GRSB19] or [CM99]. Here, one needs resort to the quadratic residuosity assumption [GRSB19] or the discrete logarithm assumption [CM99]. Also, using [GRSB19], one can show that $\gcd(\varphi(n), \gamma) = 1$. We can then turn this into a proof-of-knowledge by running a non-interactive version of the Fiat-Shamir identification scheme [FS87] on random public keys (e.g., sampled via the random oracle). This proves knowledge of the factorization of n, i.e., of p and q. Similar techniques can be used to prove knowledge of the random coins used to generate ck.

3.7 Rai-Choo Blind Signatures

In the final section of this chapter on blind signatures, we present our most practical construction, namely, the Rai-Choo blind signature scheme [HLW23a]. It is based on the CDH assumption and we present it in the type-3 pairing setting. We can view Rai-Choo as a significant improvement over our construction in Section 3.4. Namely, it no longer requires the Signer to keep a state, it is more efficient in terms of communication and computation, and it is round-optimal. In addition to our basic scheme, we show how to batch multiple signing interactions and how to extend the construction to the partially blind setting.

3.7.1 Basic Construction

Let PGGen be a bilinear group generation algorithm that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. As in Section 3.4, we assume that these system parameters are known to all algorithms, although they should formally be part of the public key. As explained in Section 3.4, these parameters are standardized and their correctness can be verified efficiently in practice, and for readability it is preferable to omit them from the public key. Our scheme $\mathsf{BS}_R = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ is parameterized by integers $K = K(\lambda)$ and $N(\lambda) \in \mathbb{N}$. Notably, these do not depend on the number of previous interactions. We only need that N^{-K} is negligible in λ. The scheme makes use of random oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu\colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_\alpha\colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}_{cc}\colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$.

Key Rerandomization. Our scheme makes use of an algorithm ReRa, that takes as input tuples $(\mathsf{pk}_i, h_i)_{i\in[K]}$ and an element $\bar\sigma \in \mathbb{G}_1$, where $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}) \in \mathbb{G}_1 \times \mathbb{G}_2$, and $h_i \in \mathbb{G}_1$ for all $i \in [K]$. The algorithm is as follows:

1. Choose $r_1, \dots, r_{K-1} \xleftarrow{\$} \mathbb{Z}_p$ and set $r_K := -\sum_{i=1}^{K-1} r_i$.
2. For all $i \in [K]$, set $\mathsf{pk}'_i := \left(\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2}\right) := \left(\mathsf{pk}_{i,1} \cdot g_1^{r_i}, \mathsf{pk}_{i,2} \cdot g_2^{r_i}\right)$.
3. Set $\bar\sigma' := \bar\sigma \cdot \prod_{i=1}^{K} h_i^{r_i}$ and return $((\mathsf{pk}'_i)_{i\in[K]}, \bar\sigma')$.

It is easy to see that $\prod_{i\in K} \mathsf{pk}_{i,j} = \prod_{i\in K} \mathsf{pk}'_{i,j}$ for both $j \in \{1,2\}$. Further, if we assume that the inputs satisfy $e\left(\bar\sigma, g_2\right) = \prod_{i=1}^{K} e\left(h_i, \mathsf{pk}_{i,2}\right)$ and $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$, then the outputs satisfy $e\left(\bar\sigma', g\right) = \prod_{i=1}^{K} e\left(h_i, \mathsf{pk}'_{i,2}\right)$ and $e\left(\mathsf{pk}'_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}'_{i,2}\right)$ for all $i \in [K]$. Additionally, the output does not reveal anything about the input, except what is already revealed by these properties. We make this more formal in Lemma 3.9 when we analyze the blindness property of our scheme.

Key Generation. To generate keys, algorithm $\mathsf{Gen}(1^\lambda)$ does the following:

1. Sample $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_p$, set $\mathsf{pk}_1 := g_1^{\mathsf{sk}}$ and $\mathsf{pk}_2 := g_2^{\mathsf{sk}}$.
2. Return public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2)$ and secret key sk.

Signature Issuing. The algorithms S, U and their interaction are given in Figures 3.6 and 3.7.

Verification. The resulting signature $\sigma := ((\mathsf{pk}_i, \varphi_i)_{i=1}^{K-1}), \varphi_K, \bar\sigma)$ for a message m is verified by algorithm $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$ as follows:

1. Write $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ for each $i \in [K-1]$.
2. Compute $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$.
3. If there is an $i \in [K]$ with $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$, return 0.
4. For each instance $i \in [K]$, compute $\mu_i := \mathsf{H}_\mu(\mathsf{m}, \varphi_i)$.
5. Return 1 if and only if $e\left(\bar\sigma, g_2\right) = \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_i), \mathsf{pk}_{i,2}\right)$.

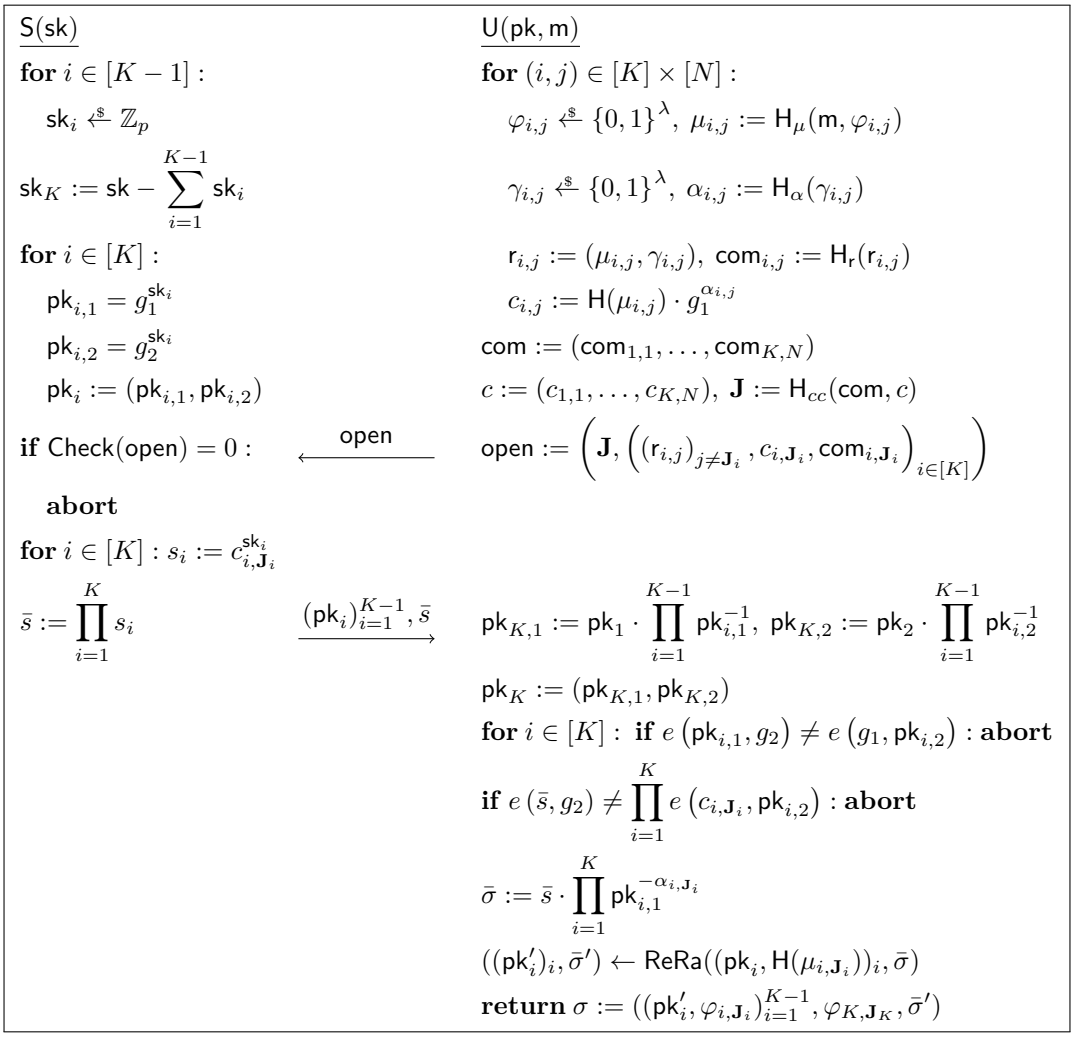

Figure 3.6: Signature issuing protocol of the blind signature scheme BS_R, where algorithm Check is defined in Figure 3.7.

Alg $\mathsf{Check}\left(\mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, c_{i,\mathbf{J}_i}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i \in [K]}\right)\right)$

```
01 for i ∈ [K] :
02   for j ∈ [N] \ {J_i} :
03     parse (μ_{i,j}, γ_{i,j}) := r_{i,j}, (μ_{i,j}, γ_{i,j}) ∈ {0,1}^λ × {0,1}^λ
04     α_{i,j} := H_α(γ_{i,j}),  c_{i,j} := H(μ_{i,j}) · g_1^{α_{i,j}},  com_{i,j} := H_r(r_{i,j})
05 com := (com_{1,1}, ..., com_{K,N}),  c := (c_{1,1}, ..., c_{K,N})
06 if J ≠ H_cc(com, c) : return 0
07 return 1
```

Figure 3.7: Algorithm Check used in the signature issuing protocol of blind signature scheme BS_R.

Analysis. Completeness of the scheme follows by inspection. We show blindness and one-more unforgeability. Before we give the proof of blindness, we show a lemma that is needed. Intuitively, it states that algorithm ReRa perfectly rerandomizes the key shares.

Lemma 3.9. *For any* $\mathsf{pk}_1 \in \mathbb{G}_1$ *and* $\mathsf{pk}_{i,1} \in \mathbb{G}_1, i \in [K]$ *such that* $\prod_{i=1}^{K} \mathsf{pk}_{i,1} = \mathsf{pk}$*, the following distributions* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *are identical:*

$$\mathcal{D}_1 := \left\{ \left(\mathsf{pk}_1, (\mathsf{pk}_{i,1})_{i\in[K]}, (\mathsf{pk}'_{i,1})_{i\in[K]}\right) \;\middle|\; \begin{array}{l} r_1, \ldots, r_{K-1} \xleftarrow{\$} \mathbb{Z}_p,\ r_K := -\sum_{i=1}^{K-1} r_i \\ \forall i \in [K] : \mathsf{pk}'_{i,1} := \mathsf{pk}_{i,1} \cdot g_1^{r_i} \end{array} \right\}$$

$$\mathcal{D}_2 := \left\{ \left(\mathsf{pk}_1, (\mathsf{pk}_{i,1})_{i\in[K]}, (\mathsf{pk}'_{i,1})_{i\in[K]}\right) \;\middle|\; \begin{array}{l} \forall i \in [K] : \mathsf{pk}'_{i,1} \xleftarrow{\$} \mathbb{G} \\ \mathsf{pk}'_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,1} \end{array} \right\}$$

Proof. It is sufficient to look at the distributions in terms of their exponents. To this end, let $\mathsf{pk} = g_1^x$, $\mathsf{pk}_{i,1} = g_1^{x_i}$, and $\mathsf{pk}'_{i,1} = g_1^{x'_i}$ for all $i \in [K]$. Consider the homomorphism $f : \mathbb{Z}_p^K \to \mathbb{Z}_p$ with $f(\mathbf{y}) := (1, \ldots, 1) \cdot \mathbf{y}$. Note that in distribution $\mathcal{D}_2$, the vector $\mathbf{x}' = (x'_1, \ldots, x'_K)^t$ is distributed uniformly over all vectors in $f^{-1}(x)$. Further, note that in distribution $\mathcal{D}_1$, the vector $\mathbf{x}'$ is distributed as $\mathbf{x} + \mathbf{r}$, where $\mathbf{r}$ is uniform in the kernel of f. This gives us the same distribution for $\mathbf{x}'$. Therefore, the distributions are the same. □

Theorem 3.6. *Let* $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p$ *be random oracles. Then* BS_R *satisfies malicious signer blindness.*

Concretely, for any algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}_\alpha}$ *queries to* $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu, \mathsf{H}_\alpha$ *respectively, we have*

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.$$

Proof. We set $\mathsf{BS} := \mathsf{BS}_R$ and let $\mathcal{A}$ be an adversary against the blindness of BS. Our proof is presented as a sequence of games $\mathbf{G}_{i,b}$ for $i \in [8]$ and $b \in \{0,1\}$. We set $\mathbf{G}_{0,b} := \mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$. Then, our goal is bound the distinguishing advantage

$$\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \left|\Pr\left[\mathbf{G}_{0,0} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{0,1} \Rightarrow 1\right]\right|.$$

To do that, we will change our game to end up at a game $\mathbf{G}_{8,b}$ for which we have

$$\Pr\left[\mathbf{G}_{8,0} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{8,1} \Rightarrow 1\right].$$

Game $\mathbf{G}_{0,b}$: Game $\mathbf{G}_{0,b}$ is defined as $\mathbf{G}_{0,b} := \mathbf{BLIND}^{\mathcal{A}}_{b,\mathsf{BS}}(\lambda)$. We recall this game to fix some notation. First, $\mathcal{A}$ outputs a public key pk and two messages $\mathsf{m}_0, \mathsf{m}_1$. Second, $\mathcal{A}$ is run with access to two interactive oracles O_0 and O_1. These simulate $\mathsf{U}(\mathsf{pk}, \mathsf{m}_b)$ and $\mathsf{U}(\mathsf{pk}, \mathsf{m}_{1-b})$, respectively. To distinguish variables used in the two oracles, we use superscripts L and R. That is, variables with superscript L (resp. R) are part of the interaction between $\mathcal{A}$ and O_0 (resp. O_1). For example, $\mathbf{J}^L := \mathsf{H}_{cc}(\mathsf{com}^L, c^L)$ denotes the cut-and-choose vector that O_0 computes, and open^R denotes the first message that O_1 sends to $\mathcal{A}$. For descriptions with variables without a superscript, the reader should understand them as applying to both oracles.

Game $\mathbf{G}_{1,b}$: This game is as $\mathbf{G}_{0,b}$, but we let the game abort on a certain event. Namely, the game aborts if $\mathcal{A}$ ever makes a query of the form $\mathsf{H}_\mu(\cdot, \varphi_{i,j})$ for some $i \in [K]$ and $j \in [N] \setminus \{\mathbf{J}_i\}$. Note that for these values (i,j), $\mathcal{A}$ obtains no information about $\varphi_{i,j}$ throughout the entire game. Thus, the probability that a query is of this form is at most $1/2^\lambda$. A union bound over all such (i,j), the two oracles, and the random oracle queries leads to

$$\left|\Pr\left[\mathbf{G}_{0,b} \Rightarrow 1\right] - \Pr\left[\mathbf{G}_{1,b} \Rightarrow 1\right]\right| \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{2,b}$: This game is as $\mathbf{G}_{1,b}$, but with another abort event. Concretely, the game aborts if $\mathcal{A}$ ever makes a query $\mathsf{H}_{\mathsf{r}}(\mathsf{r}_{i,\mathbf{J}_i})$, or a query $\mathsf{H}_{\alpha}(\gamma_{i,\mathbf{J}_i})$ for some $i \in [K]$. Note that $\mathsf{r}_{i,\mathbf{J}_i}$ has the form $\mathsf{r}_{i,\mathbf{J}_i} = (\mu_{i,\mathbf{J}_i}, \gamma_{i,\mathbf{J}_i})$, where $\gamma_{i,\mathbf{J}_i}$ is sampled uniformly at random from $\{0,1\}^{\lambda}$. Further, $\mathcal{A}$ obtains no information about $\gamma_{i,\mathbf{J}_i}$ throughout the entire game. Therefore, taking a union bound over all instances $i \in [K]$, the two user oracles, and the random oracle queries for both random oracles H_{r} and H_{α}, we get

$$|\Pr[\mathbf{G}_{1,b} \Rightarrow 1] - \Pr[\mathbf{G}_{2,b} \Rightarrow 1]| \leq \frac{KQ_{\mathsf{H}_{\mathsf{r}}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_{\alpha}}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{3,b}$: In this game, we change how the final signatures are computed. Specifically, suppose that the user oracle does not abort due to the condition $e(\bar{s}, g_2) \neq \prod_{i=1}^{K} e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right)$ and does not abort due to condition $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$ for any $i \in [K]$. Then, in previous games, the user oracle first computed $\bar{\sigma}$, and then executed $((\mathsf{pk}'_i)_i, \bar{\sigma}') \leftarrow \mathsf{ReRa}((\mathsf{pk}_i, \mathsf{H}(\mu_{i,\mathbf{J}_i}))_i, \bar{\sigma})$. The value $\bar{\sigma}'$ is part of the final signature. In game $\mathbf{G}_{3,b}$, we instead let the user oracle run a brute-force search to compute the unique $\bar{\sigma}''$ such that $e(\bar{\sigma}'', g_2) = \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}'_{i,2}\right)$. Then, we include $\bar{\sigma}''$ in the final signature instead of $\bar{\sigma}'$. We claim that this does not change the view of $\mathcal{A}$. To see this, first note that we did not change any verification or abort condition of the user oracles. Therefore, we can first consider the case where one of the user oracles locally outputs $\perp$. In this case, $\mathcal{A}$ gets $\perp, \perp$ as its final input in both $\mathbf{G}_{2,b}$ and $\mathbf{G}_{3,b}$. It remains to analyze the case where both user oracles do not abort. We claim that $\bar{\sigma}'$ and $\bar{\sigma}''$ are the same. To see this, assume $e(\bar{s}, g_2) = \prod_{i=1}^{K} e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right)$, and multiply both sides by $\prod_{i=1}^{K} e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right)$. We obtain

$$\begin{aligned}
e(\bar{s}, g_2) \cdot \prod_{i=1}^{K} e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) &= \prod_{i=1}^{K} e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right) \cdot \prod_{i=1}^{K} e\left(\mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) \\
\implies e\left(\bar{s} \cdot \prod_{i=1}^{K} \mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i}}, g_2\right) &= \prod_{i=1}^{K} e\left(c_{i,\mathbf{J}_i}, \mathsf{pk}_{i,2}\right) \cdot e\left(g_1^{-\alpha_{i,\mathbf{J}_i}}, \mathsf{pk}_{i,2}\right) \\
\implies e(\bar{\sigma}, g_2) &= \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}_{i,2}\right),
\end{aligned}$$

where we used $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$ on the right-hand side. Using the definition of algorithm ReRa, it is easy to see that this implies

$$e(\bar{\sigma}', g_2) = \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_{i,\mathbf{J}_i}), \mathsf{pk}'_{i,2}\right).$$

By definition, $\bar{\sigma}''$ satisfies the same equation. As their is a unique solution to this equation for given $\mathsf{pk}'_{i,2}$ and $\mu_{i,\mathbf{J}_i}, i \in [K]$, we see that $\bar{\sigma}' = \bar{\sigma}''$. We have

$$\Pr[\mathbf{G}_{2,b} \Rightarrow 1] = \Pr[\mathbf{G}_{3,b} \Rightarrow 1].$$

Game $\mathbf{G}_{4,b}$: We make another change to the computation of the final signatures. Again, suppose that the user oracle does not abort. In this game $\mathbf{G}_{4,b}$, we no longer run algorithm ReRa in this case. Instead, we compute the $\mathsf{pk}'_i = (\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2})$ as a fresh sharing via

$$\begin{gathered}
\mathsf{sk}'_i \xleftarrow{\$} \mathbb{Z}_p, \;\; \mathsf{pk}'_{i,1} := g_1^{\mathsf{sk}_i}, \;\; \mathsf{pk}'_{i,2} := g_2^{\mathsf{sk}_i} \text{ for } i \in [K-1], \\
\mathsf{pk}'_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,1}, \;\; \mathsf{pk}'_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}'^{-1}_{i,2}.
\end{gathered}$$

Note that the other output $\bar{\sigma}'$ of algorithm ReRa is no longer needed due to the previous change. To analyze this change, we first argue that the $\mathsf{pk}'_{i,2}$ are uniquely determined by the $\mathsf{pk}'_{i,1}$. Namely, if the user oracle does not abort, we know that $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$, and $e\left(\mathsf{pk}_1, g_2\right) = e\left(g_1, \mathsf{pk}_2\right)$. It is easy to see that property is preserved by algorithm ReRa. That is, we have $e\left(\mathsf{pk}'_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}'_{i,2}\right)$ for all $i \in [K]$. One can verify that our new definiton of the $\mathsf{pk}'_{i,1}, \mathsf{pk}'_{i,2}$ also satisfies this. It remains to analyze the distribution of the $\mathsf{pk}'_{i,1}$. By Lemma 3.9 the distribution of the $\mathsf{pk}'_{i,1}$ stays the same. This implies that

$$\Pr\left[\mathbf{G}_{3,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{4,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{5,b}$: In game $\mathbf{G}_{5,b}$, we first sample random vectors $\hat{\mathbf{J}}^L \xleftarrow{\$} [N]^K$ and $\hat{\mathbf{J}}^R \xleftarrow{\$} [N]^K$. Then, we let the game abort, if later we do not have $\hat{\mathbf{J}}^L = \mathbf{J}^L$ and $\hat{\mathbf{J}}^R = \mathbf{J}^R$. As the view of $\mathcal{A}$ is independent of $\hat{\mathbf{J}}^L, \hat{\mathbf{J}}^R$ until a potential abort, we have

$$\Pr\left[\mathbf{G}_{5,b} \Rightarrow 1\right] = \frac{1}{N^{2K}} \cdot \Pr\left[\mathbf{G}_{4,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{6,b}$: In game $\mathbf{G}_{6,b}$, we change how the values $\mu_{i,j}$ for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ are computed. Recall that before, they were computed as $\mu_{i,j} = \mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j})$. In $\mathbf{G}_{6,b}$, we sample $\mu_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$ for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ instead. We highlight that the game still samples the values $\varphi_{i,j}$ to determine when it has to abort according to $\mathbf{G}_{1,b}$. Due to the changes introduced in $\mathbf{G}_{1,b}$ and $\mathbf{G}_{5,b}$, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$ and $\mathcal{A}$ never queries $\mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j})$, and therefore this change does not influence the view of $\mathcal{A}$. We have

$$\Pr\left[\mathbf{G}_{5,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{6,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{7,b}$: In game $\mathbf{G}_{7,b}$, we change how the values $\alpha_{i,\hat{\mathbf{J}}_i}$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ are computed for all $i \in [K]$. Concretely, in this game, $\alpha_{i,\hat{\mathbf{J}}_i}$ is sampled uniformly at random as $\alpha_{i,\hat{\mathbf{J}}_i} \xleftarrow{\$} \mathbb{Z}_p$. Further, $\mathsf{com}_{i,\hat{\mathbf{J}}_i} \xleftarrow{\$} \{0,1\}^\lambda$ is sampled uniformly at random. Assuming that the game does not abort, we argue that the view of $\mathcal{A}$ does not change. This follows directly from the changes in $\mathbf{G}_{5,b}$ and $\mathbf{G}_{2,b}$. Namely, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$ and that $\mathcal{A}$ never makes a query $\mathsf{H}_\mathsf{r}(\mathsf{r}_{i,\hat{\mathbf{J}}_i})$. We have

$$\Pr\left[\mathbf{G}_{6,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{7,b} \Rightarrow 1\right].$$

Game $\mathbf{G}_{8,b}$: In game $\mathbf{G}_{8,b}$, we change how the values $c_{i,\hat{\mathbf{J}}_i}$ for $i \in [K]$ are computed. First, recall that in the previous games, these are computed as $c_{i,\hat{\mathbf{J}}_i} = \mathsf{H}(\mu_{i,\hat{\mathbf{J}}_i}) \cdot g_1^{\alpha_{i,\hat{\mathbf{J}}_i}}$. Now, we sample it at random using $c_{i,\hat{\mathbf{J}}_i} \xleftarrow{\$} \mathbb{G}_1$. We argue indistinguishability as follows. Due to the change introduced in $\mathbf{G}_{5,b}$, we can assume that $\hat{\mathbf{J}} = \mathbf{J}$. Then, we know that in this case $\alpha_{i,\hat{\mathbf{J}}_i}$ is only used to define $c_{i,\hat{\mathbf{J}}_i}$ and nowhere else. In particular, it is not used to derive the final signatures from the interaction, due to the change introduced in $\mathbf{G}_{3,b}$, and it is not used to define $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ due to the change in $\mathbf{G}_{7,b}$. As $\alpha_{i,\hat{\mathbf{J}}_i}$ is sampled uniformly at random due to the change in $\mathbf{G}_{7,b}$, we know that $c_{i,\hat{\mathbf{J}}_i}$ is distributed uniformly at random in $\mathbf{G}_{7,b}$. This shows that

$$\Pr\left[\mathbf{G}_{7,b} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{8,b} \Rightarrow 1\right].$$

Finally, it can be observed that the view of $\mathcal{A}$ does not depend on the bit b anymore. This is because the messages $\mathsf{m}_0, \mathsf{m}_1$ are not used in the user oracles. Instead, the user oracles use random $\mu_{i,j}$, independent of the messages, for all opened sessions $j \neq \mathbf{J}_i$, and the final signatures σ_0, σ_1 that $\mathcal{A}$ gets are computed using brute-force independent of the interactions, assuming that both interactions accept. This shows that

$$\Pr\left[\mathbf{G}_{8,0} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{8,1} \Rightarrow 1\right].$$

To conclude, we upper bound $\mathsf{Adv}^{\mathsf{blind}}_{\mathcal{A},\mathsf{BS}}(\lambda) = |\Pr[\mathbf{G}_{0,0} \Rightarrow 1] - \Pr[\mathbf{G}_{0,1} \Rightarrow 1]|$ by

$$\begin{aligned}
&|\Pr[\mathbf{G}_{4,0} \Rightarrow 1] - \Pr[\mathbf{G}_{4,1} \Rightarrow 1]| + 2\left(\frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-1}}\right)\\
&= N^{2K}\,|\Pr[\mathbf{G}_{5,0} \Rightarrow 1] - \Pr[\mathbf{G}_{5,1} \Rightarrow 1]| + \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}\\
&= N^{2K}\,|\Pr[\mathbf{G}_{8,0} \Rightarrow 1] - \Pr[\mathbf{G}_{8,1} \Rightarrow 1]| + \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}\\
&= \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.
\end{aligned}$$

□

Theorem 3.7. *Let* $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$*, and* $\mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$*, and* $\mathsf{H} \colon \{0,1\}^* \to \mathbb{G}$ *be random oracles. If* CDH *assumption holds relative to* PGGen*, then* BS_R *is one-more unforgeable.*

Concretely, for any polynomial ℓ *and any PPT algorithm* $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}_{cc}}, Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}}$ *queries to* $\mathsf{H}_{cc}, \mathsf{H}_\mathsf{r}, \mathsf{H}_\mu, \mathsf{H}$ *respectively, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\ell\text{-OMUF}}_{\mathcal{A},\mathsf{BS}_R}(\lambda) \leq \frac{Q^2_{\mathsf{H}_\mu} + Q^2_{\mathsf{H}_\mathsf{r}} + Q_{\mathsf{H}_\mathsf{r}} Q_{\mathsf{H}_{cc}} + Q_{\mathsf{H}} Q_{\mathsf{H}_\mu}}{2^\lambda} + \frac{\ell}{N^K} + 4\ell \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Proof. We set $\mathsf{BS} := \mathsf{BS}_R$ and let $\mathcal{A}$ be an adversary against the one-more unforgeability of BS. We show the statement by presenting a sequence of games. Before we go into detail, we explain the overall strategy of the proof. In our final step, we give a reduction that breaks the CDH assumption. This reduction works similar to the reduction for the BLS signature scheme [BLS01]. Namely, it embeds one part of the CDH instance in the public key, and one part in some of the random oracle queries for oracle H. In the first part of our proof, we prepare simulation of the signer oracle without using the secret key. Here, the strategy is to extract the User's randomness using the cut-and-choose technique. With overwhelming probability, in a fixed interaction, we can extract the randomness for one of the K instances, say instance i^*. Then, we compute the public key shares pk_i in a way that allows us to know all corresponding secret keys except sk_{i^*}. For instance i^*, we can simulate the signing oracle by programming random oracle H. In the second part of our proof, we prepare the extraction of the CDH solution from the forgery that $\mathcal{A}$ returns. Here, it is essential that the scheme uses random oracle H_μ to compute commitments $\mu_{i,j}$. This allows us to embed the part of the CDH input in H in a consistent way. We will now proceed more formally.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is the real one-more unforgeability game, i.e., $\mathbf{G}_0 := \ell\text{-}\mathbf{OMUF}^{\mathcal{A}}_{\mathsf{BS}}$. Let us recall this game. First, the game samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$. Then, $\mathcal{A}$ is executed on input pk, and gets concurrent access to signer oracle O, simulating $\mathsf{S}(\mathsf{sk})$. Additionally, $\mathcal{A}$ gets access to random oracles $\mathsf{H}, \mathsf{H}_\mu, \mathsf{H}_\mathsf{r}, \mathsf{H}_{cc}$. These are simulated by the game in the standard lazy way. Finally, $\mathcal{A}$ outputs pairs $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$. Denote the number of completed interactions (i.e., interactions in which O sent $\bar{s}$ to $\mathcal{A}$) by ℓ. If all m_i are distinct, all σ_i are valid signatures for m_i with respect to pk, and $k > \ell$, the game outputs 1. By definition, we have

$$\mathsf{Adv}^{\ell\text{-OMUF}}_{\mathcal{A},\mathsf{BS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Game $\mathbf{G}_1$: Game $\mathbf{G}_1$ is as $\mathbf{G}_0$, but it aborts if a collision for one of the random oracles $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu$ occurs. More precisely, let $* \in \{\mathsf{r}, \mu\}$ and consider a query $\mathsf{H}_*(x)$ for which the hash value is not yet defined. The game samples $\mathsf{H}_*(x)$ as in game $\mathbf{G}_0$. Then, the game aborts if there is another $x' \neq x$ such that $\mathsf{H}_*(x')$ is already defined and $\mathsf{H}_*(x) = \mathsf{H}_*(x')$. As the outputs of H_* are sampled uniformly from $\{0,1\}^\lambda$, we can use a union bound over all pairs of queries and get

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mu}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda}.$$

Game $\mathbf{G}_2$: Game $\mathbf{G}_2$ is as game $\mathbf{G}_1$, but we introduce a bad event and let the game abort if this bad event occurs. Concretely, consider any fixed query to oracle H_{cc} of the form $\mathsf{H}_{cc}(\mathsf{com}, c) = \mathbf{J}$ for $\mathsf{com} = (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$ and $c = (c_{1,1}, \ldots, c_{K,N})$. For such queries and all $(i, j) \in [K] \times [N]$, the game now tries to extract values $\bar{\mathsf{r}}_{i,j}$ such that $\mathsf{com}_{i,j} = \mathsf{H}_{\mathsf{r}}(\bar{\mathsf{r}}_{i,j})$. To do that, it searches through the random oracle queries for random oracle H_{r}. For those (i, j) for which such a value can not be extracted, we write $\bar{\mathsf{r}}_{i,j} = \bot$. Due to the change introduced in $\mathbf{G}_1$, there can be at most one extracted value for each (i, j). The game now aborts, if in such a query, there is some $(i, j) \in [K] \times [N]$ such that $\bar{\mathsf{r}}_{i,j} = \bot$, but later oracle H_{r} is queried and returns $\mathsf{com}_{i,j}$. Clearly, for a fixed pair of queries to H_{cc} and H_{r}, respectively, this bad event can only with probability $1/2^{\lambda}$. By a union bound we get

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}_{\mathsf{r}}} Q_{\mathsf{H}_{cc}}}{2^{\lambda}}.$$

Before we continue, we summarize what we established so far and introduce some terminology. For that, we fix an interaction between $\mathcal{A}$ and the signer oracle O. Consider the first message

$$\mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, c_{i,\mathbf{J}_i}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i \in [K]}\right)$$

that is sent by $\mathcal{A}$. Recall that after receiving this message, algorithm Check uses open to compute values $\mathsf{com} = (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$ and $c = (c_{1,1}, \ldots, c_{K,N})$. Then, it also checks if $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$. Also, consider the values $\bar{\mathsf{r}}_{i,j}$ related to the query $\mathsf{H}_{cc}(\mathsf{com}, c)$, as defined in $\mathbf{G}_2$. Assuming Check outputs 1 (i.e., $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$), we make two observations for any instance $i \in [K]$.

1. If for some $j \in [N]$ we have $\bar{\mathsf{r}}_{i,j} = \bot$, then $j = \mathbf{J}_i$. This is due to the bad event introduced in $\mathbf{G}_2$.
2. If for some $j \in [N]$ we have $\bar{\mathsf{r}}_{i,j} = (\mu, \gamma) \neq \bot$ but $c_{i,j} \neq \mathsf{H}(\mu) \cdot g_1^{\alpha}$ for $\alpha := \mathsf{H}_{\alpha}(\gamma)$, then $\mathbf{J}_i = j$. This is because we ruled out collisions for H_{r} in $\mathbf{G}_1$. Namely, as there are no collisions, we know that $\bar{\mathsf{r}}_{i,j} = \mathsf{r}_{i,j}$ for all $j \neq \mathbf{J}_i$. Therefore, $c_{i,j} = \mathsf{H}(\mu) \cdot g_1^{\alpha}$ by definition of Check.

If one of these two events occurs for some i, we say that there is a *successful cheat in instance i*. Note that the game can efficiently check if there is a successful cheat in an instance once it received open. Also note that the values $\bar{\mathsf{r}}_{i,j}$ are fixed in the moment $\mathcal{A}$ queries $\mathsf{H}_{cc}(\mathsf{com}, c)$ for the first time. In particular, they are fixed before $\mathcal{A}$ obtains any information about the uniformly random $\mathbf{J} = \mathsf{H}_{cc}(\mathsf{com}, c)$. Therefore, using the two observations above, the probability of a successful cheat in instance i is at most $1/N$. Further, as the components of $\mathbf{J}$ are sampled independently, the probability that there is a successful cheat in all K instances (in this fixed interaction) is at most $1/N^K$.

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, we introduce another abort. Namely, the game aborts, if in some interaction between $\mathcal{A}$ and the signer oracle O, there is a successful cheat in every instance $i \in [K]$, and that interaction is completed. By the discussion above, we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \frac{\ell}{N^K}.$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, we change the way the signer oracle computes the shares sk_i. Recall that before, these were computed as

$$\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p \text{ for } i \in [K-1], \;\; \mathsf{sk}_K := \mathsf{sk} - \sum_{i=1}^{K-1} \mathsf{sk}_i.$$

Then, the corresponding public key shares were computed as $\mathsf{pk}_i = (g_1^{\mathsf{sk}_i}, g_2^{\mathsf{sk}_i})$ for all $i \in [K]$. In game $\mathbf{G}_4$, the game instead defines the sk_i after it received the first message open from $\mathcal{A}$ in the following way. If Check outputs 0 or there is a successful cheat in every instance, the game behaves as before

(i.e., it aborts the interaction, or the entire execution). Otherwise, let $i^* \in [K]$ be the first instance in which there is no successful cheat. Then, the game computes

$$\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p \text{ for } i \in [K] \setminus \{i^*\}, \;\; \mathsf{sk}_{i^*} := \mathsf{sk} - \sum_{i \in [K] \setminus \{i^*\}} \mathsf{sk}_i.$$

The game defines pk_i for all $i \in [K]$ as before. It is clear that this change is only conceptual, as a uniformly random additive sharing of sk is computed in both $\mathbf{G}_3$ and $\mathbf{G}_4$. Therefore, we have

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_4 \Rightarrow 1].$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we introduce an abort related to the random oracles H and H_μ. Namely, the game aborts if the following occurs. The adversary $\mathcal{A}$ first queries $\mathsf{H}(\mu)$ for some $\mu \in \{0,1\}^*$, and after that a hash value $\mathsf{H}_\mu(x)$ is defined for some $x \in \{0,1\}^*$, and we have $\mathsf{H}_\mu(x) = \mu$. Clearly, once μ is fixed, the probability that a previously undefined hash value $\mathsf{H}_\mu(x)$ is equal to μ is at most $1/2^\lambda$. Therefore, we can use a union bound over the random oracle queries and get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{Q_{\mathsf{H}} Q_{\mathsf{H}_\mu}}{2^\lambda}.$$

Game $\mathbf{G}_6$: In this game, we introduce a purely conceptual change. To do that, we introduce maps $b[\cdot]$ and $\hat{b}[\cdot]$. Then, on a query $\mathsf{H}_\mu(\mathsf{m}, \varphi)$ for which the hash value is not yet defined, the game samples bit $\hat{b}[\mathsf{m}] \in \{0,1\}$ from a Bernoulli distribution, such that the probability that $\hat{b}[\mathsf{m}] = 1$ is $1/(\ell+1)$. Additionally, on a query $\mathsf{H}(\mu)$ for which the hash value is not yet defined, the game first searches for a previous query $(\mathsf{m}_\mu, \varphi)$ to H_μ such that $\mathsf{H}_\mu(\mathsf{m}_\mu, \varphi) = \mu$. Then, it sets $b[\mu] := \hat{b}[\mathsf{m}_\mu]$. If no such query can be found, it sets $b[\mu] := 0$. Note that due to the change in $\mathbf{G}_1$, the game can find at most one such query and m_μ is well defined. The view of $\mathcal{A}$ does not change, and we have

$$\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_7$: In this game, we introduce an initially empty set $\mathcal{L}$ and an abort related to it. In each interaction between $\mathcal{A}$ and the signer oracle O, the game simulates the oracle as in $\mathbf{G}_6$. Additionally, if the game has to provide the final message $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$, then we know that Check output 1 and the game did not abort. Therefore, there is at least one instance $i^* \in [K]$ such that $\mathcal{A}$ did not cheat successfully in instance i^*. Fix the first such instance. This means that the game could extract $\bar{\mathsf{r}}_{i^*, \mathbf{J}_{i^*}} = (\mu, \gamma)$ before (see the discussion after $\mathbf{G}_2$). In game $\mathbf{G}_7$, the game tries to extract m_μ as defined in $\mathbf{G}_6$ from μ using H_μ, and inserts (μ, m_μ) into set $\mathcal{L}$ if it could extract. Also, the game aborts if $b[\mu] = 1$. Otherwise, it computes and sends $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$ as before. We highlight that the size of $\mathcal{L}$ is at most the number of completed interactions ℓ. Next, consider the final output $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$ of $\mathcal{A}$, write $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, and set $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$. If $\mathcal{A}$ is successful, we know that $k > \ell$. Therefore, by the pigeonhole principle, there is at least one $(\tilde{r}, \tilde{i}) \in [k] \times [K]$ such that $(\mu_{\tilde{r},\tilde{i}}, \mathsf{m}_{\tilde{r}}) \notin \mathcal{L}$. Game $\mathbf{G}_7$ finds the first such $\mu_{\tilde{r},\tilde{i}}$, sets $\mu^* := \mu_{\tilde{r},\tilde{i}}$ and aborts if $b[\mu^*] = 0$. Note that we can assume that $b[\mu^*]$ is defined, as verification of $\mathcal{A}$'s output involves computing $\mathsf{H}(\mu^*)$. For the sake of analysis, $\mathbf{G}_7$ also appends further entries of the form (μ, m_μ) to $\mathcal{L}$ such that $|\mathcal{L}| = \ell$ and all entries in $\mathcal{L} \cup \{\mu^*\}$ have distinct components m_μ. It queries $\mathsf{H}(\mu)$ for all $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$. Then, it aborts if for some $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$ it holds that $b[\mu] = 1$.

To analyze the change we introduced, note that $\mathbf{G}_6$ and $\mathbf{G}_7$ only differ if $b[\mu^*] = 0$ or $b[\mu] = 1$ for some $(\mu, \mathsf{m}_\mu) \in \mathcal{L}$. This is because if the game could not extract m_μ in some interaction, then due to the changes in $\mathbf{G}_5$ and $\mathbf{G}_6$, we know that $b[\mu] = 0$. The view of $\mathcal{A}$ is independent of these bits until a potential abort occurs. This implies that

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1] \cdot \Pr[b[\mu^*] = 1 \wedge \forall (\mu, \mathsf{m}_\mu) \in \mathcal{L} : b[\mu] = 0].$$

By definition of the bits $b[\cdot]$, and the change in $\mathbf{G}_5$, we can rewrite the latter term in the product as

$$\Pr\left[\hat{b}[\mathsf{m}_{\tilde{r}}] = 1 \wedge \forall(\mu, \mathsf{m}_\mu) \in \mathcal{L} : \hat{b}[\mathsf{m}_\mu] = 0\right] = \frac{1}{\ell+1}\left(1 - \frac{1}{\ell+1}\right)^{\ell} = \frac{1}{\ell}\left(1 - \frac{1}{\ell+1}\right)^{\ell+1} \geq \frac{1}{4\ell},$$

where we used the fact $(1-1/x)^x \geq 1/4$ for all $x \geq 2$, and that all bits $\hat{b}[\cdot]$ are independent. Thus, we have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{4\ell} \cdot \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_8$: In this game, we change how random oracle H is simulated. Namely, in the beginning of the game, the game samples $Y \xleftarrow{\$} \mathbb{G}_1$ and initiates a map $t[\cdot]$. Then, on a query $\mathsf{H}(\mu)$ for which the hash value is not yet defined, the game first determines bit $b[\mu]$ as before. Then, it samples $t[\mu] \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{H}(\mu) := Y^{b[\mu]} \cdot g_1^{t[\mu]}$. Clearly, all hash values are still uniformly random and independent. Therefore, we have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_8 \Rightarrow 1].$$

Game $\mathbf{G}_9$: In this game, we change how the signing oracle computes public keys $(\mathsf{pk}_i)_i$ and the values $s_i, i \in [K]$ used to compute the final message $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$. Consider an interaction between $\mathcal{A}$ and the signer oracle and recall the definition of the instance i^* as in game $\mathbf{G}_4$. This is the first instance for which there is no successful cheat in this interaction, i.e., $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\mu, \gamma) \neq \bot$ could be extracted and $c_{i^*,\mathbf{J}_{i^*}} = \mathsf{H}(\mu) \cdot g_1^{\alpha}$ for $\alpha := \mathsf{H}_\alpha(\gamma)$. In $\mathbf{G}_9$, the public keys $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ are computed via

$$\mathsf{pk}_{i,1} = g_1^{\mathsf{sk}_i} \text{ for } i \in [K] \setminus \{i^*\}, \quad \mathsf{pk}_{i^*,1} := \mathsf{pk}_1 \cdot \prod_{i \in [K]\setminus\{i^*\}} \mathsf{pk}_{i,1}^{-1},$$
$$\mathsf{pk}_{i,2} = g_2^{\mathsf{sk}_i} \text{ for } i \in [K] \setminus \{i^*\}, \quad \mathsf{pk}_{i^*,2} := \mathsf{pk}_2 \cdot \prod_{i \in [K]\setminus\{i^*\}} \mathsf{pk}_{i,2}^{-1}.$$

Further, due to the aborts introduced in previous games, we know that the game only has to send $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}$ if i^* is defined and $b[\mu] = 0$, where μ is as above. In this case, game $\mathbf{G}_8$ would compute

$$s_{i^*} = c_{i^*,\mathbf{J}_{i^*}}^{\mathsf{sk}_{i^*}} = \mathsf{H}(\mu)^{\mathsf{sk}_{i^*}} \cdot g_1^{\alpha \cdot \mathsf{sk}_{i^*}} = \left(Y^{b[\mu]} \cdot g_1^{t[\mu]}\right)^{\mathsf{sk}_{i^*}} \cdot \mathsf{pk}_{i^*,1}^{\alpha} = \mathsf{pk}_{i^*,1}^{\alpha + t[\mu]}.$$

Game $\mathbf{G}_9$ computes s_{i^*} directly as $\mathsf{pk}_{i^*,1}^{\alpha+t[\mu]}$, and all other $s_i,\ i \neq i^*$ as before using sk_i. Both changes are only conceptual and allow the game to provide the signer oracle without using the secret key sk at all. We have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Finally, we give a reduction $\mathcal{B}$ against the CDH assumption that is successful if $\mathbf{G}_9$ outputs 1. We argue that

$$\Pr[\mathbf{G}_9 \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

The reduction $\mathcal{B}$ is as follows.

- Reduction $\mathcal{B}$ gets as input $g_1, g_2, e, p, X_1, Y \in \mathbb{G}_1$, and $X_2 \in \mathbb{G}_2$. It sets $\mathsf{pk}_1 := X_1, \mathsf{pk}_2 := X_2$ and uses Y as explained in $\mathbf{G}_8$.
- Reduction $\mathcal{B}$ simulates $\mathbf{G}_9$ for $\mathcal{A}$. Note that it can do that efficiently, as sk is not needed.
- When $\mathcal{A}$ terminates with its final output $(\mathsf{m}_1, \sigma_1), \ldots, (\mathsf{m}_k, \sigma_k)$, the reduction $\mathcal{B}$ writes $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, $\mathsf{pk}_{r,i} = (\mathsf{pk}_{r,i,1}, \mathsf{pk}_{r,i,2})$, sets $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$ and $\mathsf{pk}_{r,K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{r,i,1}^{-1}$ and $\mathsf{pk}_{r,K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{r,i,2}^{-1}$ for all $r \in [k]$.

It performs all checks as in $\mathbf{G}_9$. If $\mathbf{G}_9$ outputs 1, we know that $\mathcal{B}$ defined $\mu^* := \mu_{\tilde{r},\tilde{i}}$ as $\mathbf{G}_9$ does. Then, $\mathcal{B}$ outputs

$$Z := \bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mu_{\tilde{r},i}]}.$$

It is clear that $\mathcal{B}$ perfectly simulates $\mathbf{G}_9$ and the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. Thus, it remains to argue that if $\mathbf{G}_9$ outputs 1, the Z is a valid CDH solution. To this end, assume that $\mathbf{G}_9$ outputs 1. It is sufficient to show that $e(Y, X_2) = e(Z, g_2)$.

First, note that due to the abort that we introduced in $\mathbf{G}_5$, we know that for all $i \in [K]$, the query $\mathsf{H}_\mu(\mathsf{m}_{\tilde{r}}, \varphi_{\tilde{r},i})$ was made before bit $b[\mu_{\tilde{r},i}]$ was defined. Therefore, due to the change in $\mathbf{G}_6$, we obtain for all $i \in [K]$

$$b[\mu_{\tilde{r},i}] = \hat{b}[\mathsf{m}_{\tilde{r}}] = b[\mu_{\tilde{r},\tilde{r}}] = b[\mu^*] = 1.$$

Second, we know that we have $\prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,2} = X_2$, and by definition of the verification algorithm we have

$$\begin{aligned} e(\bar{\sigma}_{\tilde{r}}, g_2) &= \prod_{i=1}^{K} e\left(\mathsf{H}(\mu_{r,i}), \mathsf{pk}_{\tilde{r},i,2}\right) = \prod_{i=1}^{K} e\left(Y \cdot g^{t[\mu_{\tilde{r},i}]}, \mathsf{pk}_{\tilde{r},i,2}\right) \\ &= \prod_{i=1}^{K} e\left(Y, \mathsf{pk}_{\tilde{r},i,2}\right) \cdot e\left(\mathsf{pk}_{\tilde{r},i,1}^{t[\mu_{\tilde{r},i}]}, g_2\right) = e(Y, X_2) \cdot e\left(\prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{t[\mu_{\tilde{r},i}]}, g_2\right). \end{aligned}$$

In the third equation we used $e\left(\mathsf{pk}_{\tilde{r},i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{\tilde{r},i,2}\right)$ for all $i \in [K]$. This implies that

$$e(Z, g_2) = e\left(\bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mu_{\tilde{r},i}]}, g_2\right) = e(Y, X_2).$$

□

3.7.2 Extension: Partial Blindness and Batching

We present a batching technique for our blind signature scheme, which leads to a significant efficiency improvement in terms of communication. At the same time, we show how to make our scheme partially blind. As for BS_R, we let PGGen be a bilinear group generation algorithm that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$. Our scheme $\mathsf{BPBS}_R = (\mathsf{Gen}, \mathsf{S}, \mathsf{U}, \mathsf{Ver})$ is parameterized by integers $K = K(\lambda)$ and $N(\lambda) \in \mathbb{N}$, where we need that N^{-K} is negligible in λ. We assume that the space $\mathcal{I}$ of public information strings contains bitstrings of bounded length[11]. The scheme makes use of random oracles $\mathsf{H}_{\mathsf{r}}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda, \mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p, \mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} : \{0,1\}^* \to \mathbb{G}_1$. We verbally describe the signature issuing protocol (S, U) and verification of scheme BPBS_R. Key generation (algorithm Gen) is exactly as in BS_R.

Signature Issuing. The interactive signature issuing protocol between algorithms $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]})$ and $\mathsf{U}(\mathsf{pk}, L, (\mathsf{m}_l, \mathsf{info}_l)_{l\in[L]})$ is given as follows.

1. User U does the following.

   (a) *Preparation.* First, for each instance $i \in [K]$ and session $j \in [N]$, U commits to all L messages via

   $$\varphi_{i,j,l} \xleftarrow{\$} \{0,1\}^\lambda, \;\; \mu_{i,j,l} := \mathsf{H}_\mu(\mathsf{m}_l, \varphi_{i,j,l}) \text{ for all } (i,j,l) \in [K] \times [N] \times [L].$$

[11] This is without loss of generality, using a collision-resistant hash function.

(b) *Commitments.* Next, for each instance $i \in [K]$ and session $j \in [N]$, U samples a seed $\gamma_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$. It then defines

$$\mathsf{r}_{i,j} := (\gamma_{i,j}, \mu_{i,j,1}, \ldots, \mu_{i,j,L}), \quad \mathsf{com}_{i,j} := \mathsf{H}_\mathsf{r}(\mathsf{r}_{i,j}) \text{ for all } (i,j) \in [K] \times [N].$$

Then, U sets $\mathsf{com} := (\mathsf{com}_{1,1}, \ldots, \mathsf{com}_{K,N})$.

(c) *Challenges.* Now, U derives randomness $\alpha_{i,j,l}$ and computes challenges $c_{i,j,l}$ via $\alpha_{i,j,l} := \mathsf{H}_\alpha(\gamma_{i,j}, l)$ and $c_{i,j,l} := \mathsf{H}(\mathsf{info}_l, \mu_{i,j,l}) \cdot g_1^{\alpha_{i,j,l}}$ for all $(i,j,l) \in [K] \times [N] \times [L]$. Then, U sets $c := (c_{1,1,1}, \ldots, c_{K,N,L})$.

(d) *Cut-and-Choose.* Next, U derives a cut-and-choose vector $\mathbf{J} \in [N]^K$ as $\mathbf{J} := \mathsf{H}_{cc}(\mathsf{com}, c)$. It then defines an opening

$$\mathsf{open} := \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j \neq \mathbf{J}_i}, (c_{i,\mathbf{J}_i,l})_{l \in [L]}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i \in [K]}\right).$$

Finally, U sends open to S.

2. Signer S does the following.

(a) *Key Sharing.* First, S samples $\mathsf{sk}_i \xleftarrow{\$} \mathbb{Z}_p$ for $i \in [K-1]$. It computes $\mathsf{sk}_K := \mathsf{sk} - \sum_{i=1}^{K-1} \mathsf{sk}_i$ and $\mathsf{pk}_i := (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}) := (g_1^{\mathsf{sk}_i}, g_2^{\mathsf{sk}_i})$ for all $i \in [K]$.

(b) *Cut-and-Choose Verification.* To verify the opening, S runs $\mathsf{Check}(L, (\mathsf{info}_l)_{l \in [L]}, \mathsf{open})$ (see Figure 3.8). If this algorithm returns 0, S aborts the interaction.

(c) *Responses.* For each instance $i \in [K]$ and each $l \in [L]$, S computes responses $s_{i,l} := c_{i,\mathbf{J}_i,l}^{\mathsf{sk}_i}$. Then, it aggregates them for each $l \in [L]$ by computing $\bar{s}_l := \prod_{i=1}^K s_{i,l}$. Finally, S sends $(\mathsf{pk}_i)_{i=1}^{K-1}, \bar{s}_1, \ldots, \bar{s}_L$ to U.

3. User U does the following.

(a) *Key Sharing Verification.* First, U recomputes key pk_K as $\mathsf{pk}_K := (\mathsf{pk}_{K,1}, \mathsf{pk}_{K,2})$ for $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$. Next, U checks validity of the pk_i by checking if $e\left(\mathsf{pk}_{i,1}, g_2\right) = e\left(g_1, \mathsf{pk}_{i,2}\right)$ for all $i \in [K]$. If any of these equations does not hold, U aborts the interaction.

(b) *Response Verification.* Then, U verifies the responses $\bar{s}_l$ by checking

$$e\left(\bar{s}_l, g_2\right) = \prod_{i=1}^K e\left(c_{i,\mathbf{J}_i,l}, \mathsf{pk}_{i,2}\right) \text{ for all } l \in [L].$$

If any of these equations does not hold, U aborts the interaction. Otherwise, it computes

$$\bar{\sigma}_l := \bar{s}_l \cdot \prod_{i=1}^K \mathsf{pk}_{i,1}^{-\alpha_{i,\mathbf{J}_i,l}} \text{ for all } l \in [L].$$

(c) *Key Rerandomization.* Next, U computes rerandomized key sharings via

$$((\mathsf{pk}'_{i,l})_i, \bar{\sigma}'_l) \leftarrow \mathsf{ReRa}((\mathsf{pk}_i, \mathsf{H}(\mathsf{info}_l, \mu_{i,\mathbf{J}_i,l}))_i, \bar{\sigma}_l) \text{ for all } l \in [L].$$

It then defines signatures

$$\sigma_l := ((\mathsf{pk}'_{i,l}, \varphi_{i,\mathbf{J}_i,l})_{i=1}^{K-1}, \varphi_{K,\mathbf{J}_K,l}, \bar{\sigma}'_l) \text{ for all } l \in [L].$$

(d) Finally, U outputs the signatures $\sigma_1, \ldots, \sigma_L$.

Alg Check $\left(L, (\mathsf{info}_l)_{l\in[L]}, \mathsf{open} = \left(\mathbf{J}, \left((\mathsf{r}_{i,j})_{j\neq\mathbf{J}_i}, (c_{i,\mathbf{J}_i,l})_{l\in[L]}, \mathsf{com}_{i,\mathbf{J}_i}\right)_{i\in[K]}\right)\right)$

```
for i ∈ [K] :
  for j ∈ [N] \ {J_i} :
    com_{i,j} := H_r(r_{i,j})
    parse (γ_{i,j}, μ_{i,j,1}, ..., μ_{i,j,L}) := r_{i,j},  (γ_{i,j}, μ_{i,j,1}, ..., μ_{i,j,L}) ∈ ({0,1}^λ)^{L+1}
    for l ∈ [L] :  α_{i,j,l} := H_α(γ_{i,j}, l),  c_{i,j,l} := H(info_l, μ_{i,j,l}) · g_1^{α_{i,j,l}}
com := (com_{1,1}, ..., com_{K,N}),  c := (c_{1,1,1}, ..., c_{K,N,L})
if J ≠ H_cc(com, c) : return 0
return 1
```

Figure 3.8: Algorithm Check used in the signature issuing protocol of batched blind signature scheme BPBS_R.

Verification. The resulting signature $\sigma := ((\mathsf{pk}_i, \varphi_i)_{i=1}^{K-1}), \varphi_K, \bar{\sigma})$ for a message m and string info is verified by algorithm $\mathsf{Ver}(\mathsf{pk}, \mathsf{info}, \mathsf{m}, \sigma)$ as follows:

1. Write $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ for each $i \in [K-1]$.
2. Compute $\mathsf{pk}_{K,1} := \mathsf{pk}_1 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,1}^{-1}$ and $\mathsf{pk}_{K,2} := \mathsf{pk}_2 \cdot \prod_{i=1}^{K-1} \mathsf{pk}_{i,2}^{-1}$.
3. If there is an $i \in [K]$ with $e\left(\mathsf{pk}_{i,1}, g_2\right) \neq e\left(g_1, \mathsf{pk}_{i,2}\right)$, return 0.
4. For each instance $i \in [K]$, compute $\mu_i := \mathsf{H}_\mu(\mathsf{m}, \varphi_i)$.
5. Return 1 if and only if
$$e\left(\bar{\sigma}, g_2\right) = \prod_{i=1}^{K} e\left(\mathsf{H}(\mathsf{info}, \mu_i), \mathsf{pk}_{i,2}\right).$$

Analysis. Completeness of the scheme follows by inspection. The proofs and concrete security bounds for blindness and one-more unforgeability are almost identical to the proofs of the corresponding theorems for the non-batched construction.

Theorem 3.8. *Let* $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$ *and* $\mathsf{H}_\alpha \colon \{0,1\}^* \to \mathbb{Z}_p$ *be random oracles. Then* BPBS_R *satisfies malicious signer batch partial blindness.*

Concretely, for any algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_\mu}, Q_{\mathsf{H}_\alpha}$ *queries to* $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu, \mathsf{H}_\alpha$ *respectively, we have*

$$\mathsf{Adv}^{\mathsf{bblind}}_{\mathcal{A},\mathsf{BPBS}_R}(\lambda) \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-2}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-2}}.$$

Proof. The proof is almost identical to the proof of Theorem 3.6, and we encourage the reader to read the proof of Theorem 3.6 first. Here, we only sketch the differences. Set $\mathsf{BPBS} := \mathsf{BPBS}_R$ and let $\mathcal{A}$ be an adversary against the batch partial blindness of BPBS. As in the proof of Theorem 3.6, we prove the statement using games $\mathbf{G}_{i,b}$ for $i \in [8]$ and $b \in \{0,1\}$ such that

$$\Pr\left[\mathbf{G}_{8,0} \Rightarrow 1\right] = \Pr\left[\mathbf{G}_{8,1} \Rightarrow 1\right].$$

Game $\mathbf{G}_{0,b}$: We set $\mathbf{G}_{0,b}$ as $\mathbf{G}_{0,b} := \mathbf{BBLIND}^{\mathcal{A}}_{b,\mathsf{BPBS}}(\lambda)$. Recall that in this game, $\mathcal{A}$ outputs a public key pk, lists $(\mathsf{info}^L_l, \mathsf{m}^L_l)_{l\in[L^L]}$ and $(\mathsf{info}^R_l, \mathsf{m}^R_l)_{l\in[L^R]}$, and indices l^L_* and l^R_*. The game outputs 0 if $\mathsf{info}^L_{l^L_*} \neq \mathsf{info}^R_{l^R_*}$. For the rest of the proof, we can assume that $\mathsf{info}^L_{l^L_*} = \mathsf{info}^R_{l^R_*}$. Then, if $b = 1$, the messages $\mathsf{m}^L_{l^L_*}$ and $\mathsf{m}^R_{l^R_*}$ are swapped. Adversary $\mathcal{A}$ gets access to oracles O_0 and O_1 simulating

$$\mathsf{U}(\mathsf{pk}, L^L, (\mathsf{m}^L_l, \mathsf{info}^L_l)_{l\in[L^L]}) \text{ and } \mathsf{U}(\mathsf{pk}, L^R, (\mathsf{m}^R_l, \mathsf{info}^R_l)_{l\in[L^R]}),$$

respectively. As in the proof of Theorem 3.6, we use superscripts L and R to distinguish variables used in these oracles. If no superscript is given, the description refers to both oracles.

Game $\mathbf{G}_{1,b}$: Game $\mathbf{G}_{1,b}$ is as $\mathbf{G}_{0,b}$, but we add an abort on a certain event. Namely, the game aborts if $\mathcal{A}$ ever queries $\mathsf{H}_\mu(\cdot, \varphi_{i,j,l_*})$ for some $i \in [K]$ and $j \in [N] \setminus \{\mathbf{J}_i\}$. As $\mathcal{A}$ obtains no information about φ_{i,j,l_*} over the entire game, we have

$$|\Pr[\mathbf{G}_{0,b} \Rightarrow 1] - \Pr[\mathbf{G}_{1,b} \Rightarrow 1]| \leq \frac{KNQ_{\mathsf{H}_\mu}}{2^{\lambda-1}}.$$

Game $\mathbf{G}_{2,b}$: Game $\mathbf{G}_{2,b}$ is as $\mathbf{G}_{1,b}$, but with another abort event. The game aborts, if $\mathcal{A}$ ever makes a query $\mathsf{H}_\mathsf{r}(\mathsf{r}_{i,\mathbf{J}_i})$, or a query $\mathsf{H}_\alpha(\gamma_{i,\mathbf{J}_i}, l_*)$ for some $i \in [K]$. As in the proof of Theorem 3.6, the probability of such an abort is negligible due to the entropy of $\gamma_{i,\mathbf{J}_i}$, and we get

$$|\Pr[\mathbf{G}_{1,b} \Rightarrow 1] - \Pr[\mathbf{G}_{2,b} \Rightarrow 1]| \leq \frac{KQ_{\mathsf{H}_\mathsf{r}}}{2^{\lambda-1}} + \frac{KQ_{\mathsf{H}_\alpha}}{2^{\lambda-1}}.$$

Games $\mathbf{G}_{3,b}$-$\mathbf{G}_{4,b}$: The changes we introduce in these games are exactly as in the proof of Theorem 3.6, but it is sufficient to apply them to the final signatures for messages $\mathsf{m}^L_{l^L_*}$ and $\mathsf{m}^R_{l^R_*}$. As in the proof of Theorem 3.6, we have

$$\Pr[\mathbf{G}_{2,b} \Rightarrow 1] = \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{5,b}$: This change is exactly as in the proof of Theorem 3.6, i.e., we let the game sample random vectors $\hat{\mathbf{J}}^L \xleftarrow{\$} [N]^K$ and $\hat{\mathbf{J}}^R \xleftarrow{\$} [N]^K$, and later the game aborts if we do not have $\hat{\mathbf{J}}^L = \mathbf{J}^L$ and $\hat{\mathbf{J}}^R = \mathbf{J}^R$. We have

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \frac{1}{N^{2K}} \cdot \Pr[\mathbf{G}_{4,b} \Rightarrow 1].$$

Game $\mathbf{G}_{6,b}$: In this game, we change how the values μ_{i,j,l_*} for $i \in [K]$ and $j \in [N] \setminus \{\hat{\mathbf{J}}_i\}$ are computed. Namely, while they were defined as $\mu_{i,j,l} := \mathsf{H}_\mu(\mathsf{m}, \varphi_{i,j,l})$ before, we now sample them at random, i.e., $\mu_{i,j,l} \xleftarrow{\$} \{0,1\}^\lambda$. We can argue using the change we introduced in $\mathbf{G}_{1,b}$, and similar to the proof of Theorem 3.6, to get

$$\Pr[\mathbf{G}_{5,b} \Rightarrow 1] = \Pr[\mathbf{G}_{6,b} \Rightarrow 1].$$

Game $\mathbf{G}_{7,b}$: We change how the values $\alpha_{i,\hat{\mathbf{J}}_i,l_*}$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i}$ are computed for all $i \in [K]$. Namely we sample $\alpha_{i,\hat{\mathbf{J}}_i,l_*} \xleftarrow{\$} \mathbb{Z}_p$ and $\mathsf{com}_{i,\hat{\mathbf{J}}_i} \xleftarrow{\$} \{0,1\}^\lambda$. We can argue using the changes in $\mathbf{G}_{5,b}$ and $\mathbf{G}_{2,b}$, and similar to the proof of Theorem 3.6, to get

$$\Pr[\mathbf{G}_{6,b} \Rightarrow 1] = \Pr[\mathbf{G}_{7,b} \Rightarrow 1].$$

Game $\mathbf{G}_{8,b}$: We change how the values $c_{i,\hat{\mathbf{J}}_i,l_*}$ are computed. Namely, we now sample these at random, i.e., $c_{i,\hat{\mathbf{J}}_i,l_*} \xleftarrow{\$} \mathbb{G}_1$. We can argue as in the proof of Theorem 3.6, to get

$$\Pr[\mathbf{G}_{7,b} \Rightarrow 1] = \Pr[\mathbf{G}_{8,b} \Rightarrow 1].$$

Finally, it can be observed that the view of $\mathcal{A}$ is independent of the bit b. The statement follows as in the proof of Theorem 3.6. □

Theorem 3.9. *Let $\mathsf{H}_\mathsf{r}, \mathsf{H}_\mu \colon \{0,1\}^* \to \{0,1\}^\lambda$, and $\mathsf{H}_{cc} \colon \{0,1\}^* \to [N]^K$, and $\mathsf{H} \colon \{0,1\}^* \to \mathbb{G}$ be random oracles. If* CDH *assumption holds relative to* PGGen*, then BPBS_R is batch one-more unforgeable.*

Concretely, for any polynomial ℓ and any PPT algorithm $\mathcal{A}$ that makes at most $Q_{\mathsf{H}_{cc}}, Q_{\mathsf{H}_\mathsf{r}}, Q_{\mathsf{H}_\mu}, Q_\mathsf{H}$ queries to $\mathsf{H}_{cc}, \mathsf{H}_\mathsf{r}, \mathsf{H}_\mu, \mathsf{H}$ respectively, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and

$$\mathsf{Adv}^{\ell\text{-BOMUF}}_{\mathcal{A},\mathsf{BPBS}_R}(\lambda) \leq \frac{Q^2_{\mathsf{H}_\mu} + Q^2_{\mathsf{H}_\mathsf{r}} + Q_{\mathsf{H}_\mathsf{r}} Q_{\mathsf{H}_{cc}} + Q_\mathsf{H} Q_{\mathsf{H}_\mu}}{2^\lambda} + \frac{\ell}{N^K} + 4\ell \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Proof. The proof is a direct generalization of the proof of Theorem 3.7. The reader should read the proof of Theorem 3.7 first. We only sketch the differences. Set $\mathsf{BPBS} := \mathsf{BPBS}_R$ and let $\mathcal{A}$ be an adversary against the batch one-more unforgeability of BPBS. As in the proof of Theorem 3.7, we prove the theorem using a sequence of games $\mathbf{G}_i, i \in [9]$.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is defined to be $\mathbf{G}_0 := \ell\text{-}\mathbf{BOMUF}^{\mathcal{A}}_{\mathsf{BPBS}}$. That is, first a pair of keys $(\mathsf{pk}, \mathsf{sk}) \leftarrow \overline{\mathsf{Gen}}(1^\lambda)$ is sampled. Then, $\mathcal{A}$ can access a signer oracle O. In each interaction this oracle takes as input a batch size $L \in \mathbb{N}$ and L strings $\mathsf{info}_1, \ldots, \mathsf{info}_L \in \mathcal{I}$, and then simulates $\mathsf{S}(\mathsf{sk}, L, (\mathsf{info}_l)_{l\in[L]})$. Further, $\mathcal{C}$ denotes the list of all tuples $(i, L, (\mathsf{info}_l)_{l\in[L]})$ such that $\mathcal{A}$ submitted batch size L and strings $(\mathsf{info}_l)_{l\in[L]}$ in the ith completed interaction with O. We have $\sum_{(i,L,(\mathsf{info}_l)_{l\in[L]})\in\mathcal{C}} L \leq \ell$. In the end, $\mathcal{A}$ outputs tuples $(\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)$. Adversary $\mathcal{A}$ wins, if there is some info^*, such that, considering only the tuples with first component $\mathsf{info}*$, all messages m_i are distinct, all signatures σ_i are valid, and there are more tuples of this form than completed interactions with this info^*. We have

$$\mathsf{Adv}^{\ell\text{-}\mathsf{BOMUF}}_{\mathcal{A},\mathsf{BPBS}}(\lambda) = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Games $\mathbf{G}_1$-$\mathbf{G}_5$: These games are exactly as in the proof of Theorem 3.7. Namely, we rule out collisions for random oracles H_μ, H_r, let the game extract values $\bar{\mathsf{r}}_{i,j}$ during queries to H_{cc}, and establish that there is at least one instance i^* per interaction, for which the adversary does not cheat successfully. Recall that this means that the game can extract $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\gamma, \mu_1, \ldots, \mu_L) \neq \bot$ and $c_{i^*,\mathbf{J}_{i^*},l} = \mathsf{H}(\mathsf{info}_l, \mu_l) \cdot g_1^{\alpha_l}$ for $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$ and all $l \in [L]$. Here, L and $(\mathsf{info}_l)_{l\in[L]}$ denote the batch size and the public strings that the adversary submitted to oracle O in this interaction. Also, in this sequence of games we change the way the secret keys sk_i are sampled, and introduce an abort related to the order of queries for H and H_μ All of this is done exactly as in the proof of Theorem 3.7. We have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{Q^2_{\mathsf{H}_\mu}}{2^\lambda} + \frac{Q^2_{\mathsf{H}_\mathsf{r}}}{2^\lambda} + \frac{Q_{\mathsf{H}_\mathsf{r}} Q_{\mathsf{H}_{cc}}}{2^\lambda} + \frac{\ell}{N^K} + \frac{Q_\mathsf{H} Q_{\mathsf{H}_\mu}}{2^\lambda}.$$

Game $\mathbf{G}_6$: Game $\mathbf{G}_6$ is as $\mathbf{G}_5$, but we introduce a conceptual change. Namely, the game now holds maps $b[\cdot]$ and $\hat{b}[\cdot]$. On a query of the form $\mathsf{H}(\mathsf{info}, \mu)$, the game first searches for a previous query $(\mathsf{m}_\mu, \varphi)$ to H_μ such that $\mathsf{H}_\mu(\mathsf{m}_\mu, \varphi) = \mu$. If no such query is found, the game sets $b[\mathsf{info}, \mu] := 0$. If such a query $(\mathsf{m}_\mu, \varphi)$ is found, and $\hat{b}[\mathsf{info}, \mathsf{m}_\mu]$ is not yet defined, the game samples $\hat{b}[\mathsf{info}, \mathsf{m}_\mu] \in \{0, 1\}$ from a Bernoulli distribution, such that the probability that $\hat{b}[\mathsf{info}, \mathsf{m}_\mu] = 1$ is $1/(\ell + 1)$. The game then sets $b[\mathsf{info}, \mu] := \hat{b}[\mathsf{info}, \mathsf{m}_\mu]$. Note that the game can find at most one m_μ for a given μ, due to the change introduced in $\mathbf{G}_1$. Clearly, the view of the adversary $\mathcal{A}$ does not change, and we have

$$\Pr[\mathbf{G}_5 \Rightarrow 1] = \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_7$: As in the proof of Theorem 3.7, we introduce an initially empty set $\mathcal{L}$ in this game, and add a new abort event. To this end, consider an interaction between the adversary $\mathcal{A}$ and the signer oracle. In this interaction, we know that there is at least one instance i^* for which the adversary does not cheat successfully. That is, the game can extract $\bar{\mathsf{r}}_{i^*,\mathbf{J}_{i^*}} = (\gamma, \mu_1, \ldots, \mu_L) \neq \bot$ and $c_{i^*,\mathbf{J}_{i^*},l} = \mathsf{H}(\mathsf{info}_l, \mu_l) \cdot g_1^{\alpha_l}$ for $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$ and all $l \in [L]$. Additionally, in $\mathbf{G}_7$, the game now tries to extract queries $(\mathsf{m}_{\mu_l}, \varphi_l)$ for all $l \in [L]$ such that $\mu_l = \mathsf{H}_\mu(\mathsf{m}_{\mu_l}, \varphi_l)$. For those $l \in [L]$ for which it can extract, it inserts $(\mathsf{info}_l, \mu_l, \mathsf{m}_{\mu_l})$ into $\mathcal{L}$. It is clear that the size of $\mathcal{L}$ is at most $\sum_{(i,L,(\mathsf{info}_l)_{l\in[L]})\in\mathcal{C}} L \leq \ell$. Additionally, if there is some $l \in [L]$ such that $b[\mathsf{info}_l, \mu_l] = 1$, the game aborts.

Further, consider the final output $(\mathsf{info}_1, \mathsf{m}_1, \sigma_1), \ldots, (\mathsf{info}_k, \mathsf{m}_k, \sigma_k)$ of $\mathcal{A}$. Write $\sigma_r = ((\mathsf{pk}_{r,i}, \varphi_{r,i})_{i=1}^{K-1}), \varphi_{r,K}, \bar{\sigma}_r)$, and set $\mu_{r,i} := \mathsf{H}_\mu(\mathsf{m}_r, \varphi_{r,i})$ for all $r \in [k], i \in [K]$. If $\mathcal{A}$ is successful, we know that there is some info^* such that $|\mathsf{Compl}[\mathsf{info}^*]| < |\mathsf{Out}[\mathsf{info}^*]|$, where

$$\mathsf{Compl}[\mathsf{info}^*] := \{(i, l_0) \mid \exists (i, L, (\mathsf{info}_l)_{l\in[L]}) \in \mathcal{C} : \mathsf{info}_{l_0} = \mathsf{info}^*\}$$
$$\mathsf{Out}[\mathsf{info}^*] := \{i \in [k] \mid \mathsf{info}_i = \mathsf{info}^*\}.$$

It is easy to see that $|\mathsf{Compl}[\mathsf{info}^*]|$ is an upper bound for the number of tuples $(\mathsf{info}, \mu, \mathsf{m}_\mu)$ in $\mathcal{L}$ with $\mathsf{info} = \mathsf{info}^*$. Therefore, by the pigeonhole principle, we know that there is at least one $(\tilde{r}, \tilde{i}) \in \mathsf{Out}[\mathsf{info}^*] \times [K]$ such that $(\mathsf{info}_{\tilde{r}}, \mu_{\tilde{r},\tilde{i}}, \mathsf{m}_{\tilde{r}}) \notin \mathcal{L}$. Here we have $\mathsf{info}_{\tilde{r}} = \mathsf{info}^*$. Game $\mathbf{G}_7$ finds the first such $(\tilde{r}, \tilde{i})$, sets $\mu^* := \mu_{\tilde{r},\tilde{i}}$ and aborts if $b[\mathsf{info}^*, \mu^*] = 0$. As in the proof of Theorem 3.7, we can assume that $b[\mathsf{info}^*, \mu^*]$ is defined. Also, as in the proof of Theorem 3.7, the game adds further entries $(\mathsf{info}, \mu, \mathsf{m}_\mu)$ to $\mathcal{L}$ such that $|\mathcal{L}| = \ell$, and aborts if $b[\mathsf{info}, \mu] = 1$.

The analysis of this change is as in the proof of Theorem 3.7. Namely, we first see that $\mathbf{G}_7$ outputs 1 if $\mathbf{G}_6$ outputs 1 and

$$b[\mathsf{info}^*, \mu^*] = 1 \wedge \forall(\mathsf{info}, \mu, \mathsf{m}_\mu) \in \mathcal{L} : b[\mathsf{info}, \mu] = 0.$$

Then, we use the same calculation and arguments as in the proof of Theorem 3.7 to get

$$\Pr[\mathbf{G}_7 \Rightarrow 1] \geq \frac{1}{4\ell} \cdot \Pr[\mathbf{G}_6 \Rightarrow 1].$$

Game $\mathbf{G}_8$: As in the proof of Theorem 3.7, we change how random oracle H is simulated. In the beginning of the game, the game samples $Y \xleftarrow{\$} \mathbb{G}_1$ and initiates an empty map $t[\cdot]$. On a query $\mathsf{H}(\mathsf{info}, \mu)$ for which the hash value is not yet defined, the game first determines $b[\mathsf{info}, \mu]$ as explained in $\mathbf{G}_6$. Then, it samples $t[\mathsf{info}, \mu] \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{H}(\mu) := Y^{b[\mathsf{info},\mu]} \cdot g_1^{t[\mathsf{info},\mu]}$. This does not change the view of the adversary. We have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_8 \Rightarrow 1].$$

Game $\mathbf{G}_9$: This change is as in the proof of Theorem 3.7. Namely, we change how the public key sharing $(\mathsf{pk}_i)_{i\in[K]}$ and the values $s_{i^*,l}$ for all $l \in [L]$ are computed in an interaction. The sharing $(\mathsf{pk}_i)_{i\in[K]}$ is computed exactly as in the proof of Theorem 3.7, and the values $s_{i^*,l}$ are now computed as $s_{i^*,l} = \mathsf{pk}_{i^*,1}^{\alpha_l + t[\mathsf{info}_l, \mu_l]}$, where $\alpha_l := \mathsf{H}_\alpha(\gamma, l)$, and $(\gamma, \mu_1, \ldots, \mu_L)$ has been extracted by the game. Due to this change, sk is no longer needed. As in the proof of Theorem 3.7, we have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Finally, as in the proof of Theorem 3.7, we can give a reduction $\mathcal{B}$ against the CDH assumption that is successful if $\mathbf{G}_9$ outputs 1. This reduction gets as input $g_1, g_2, e, p, X_1, Y \in \mathbb{G}_1$, and $X_2 \in \mathbb{G}_2$. It sets $\mathsf{pk}_1 := X_1, \mathsf{pk}_2 := X_2$ and uses Y as explained in $\mathbf{G}_8$. Then, it simulates $\mathbf{G}_9$ for $\mathcal{A}$. Finally, it outputs

$$Z := \bar{\sigma}_{\tilde{r}} \cdot \prod_{i=1}^{K} \mathsf{pk}_{\tilde{r},i,1}^{-t[\mathsf{info}^*, \mu_{\tilde{r},i}]}.$$

Analysis of $\mathcal{B}$ is as in the proof of Theorem 3.7, and we conclude with

$$\Pr[\mathbf{G}_9 \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

□

3.7.3 Concrete Parameters and Efficiency

In this section, we discuss concrete parameters and efficiency of our scheme.

Instantiating Parameters. We instantiate our scheme over the BLS12-381 curve, using SHA-256 as a hash function. It remains to determine appropriate choices for parameters K and N. To do that, we first fix some choice of N and a security level $\kappa = 128$. Then, we assume a maximum number of $\ell = 2^{30}$ signing interactions with the same key. Following the security bound, we can now set $K := \lceil (\kappa + \log \ell)/\log N \rceil + 1$. This approach leads to the instantiations

$$\text{(I)}\ K = 80,\ N = 4, \quad \text{(II)}\ K = 54,\ N = 8, \quad \text{(III)}\ K = 33,\ N = 32.$$

For these, we compute the sizes of signatures and communication in a Python script[12]. Our results are presented in Table 3.3. Let us briefly discuss the results. Especially, we want to compare the communication complexity of our scheme to the communication complexity of PI-Cut-Choo (Section 3.4). For that, we use instantiation (I) of our scheme, which has roughly 33 KB of communication. For PI-Cut-Choo, the communication is roughly $26 + 3 \cdot \log(N)$ KB. For $N = 6$, the communication size of PI-Cut-Choo already exceeds the size of Rai-Choo. Thus, even assuming that N is exactly the number of interactions, PI-Cut-Choo is only better in the first 5 interactions. This assumption does not even hold, as N is not identical to the number of interactions, see function f in Theorem 3.2. In addition, we remark that when setting parameters, one has to be very conservative about the number of signing interactions. Especially, for PI-Cut-Choo, the Signer's counter will increase quickly, because for each (even non-malicious) timeout, the counter has to be increased. Such timeouts are even more likely when the scheme requires multiple signing rounds, as PI-Cut-Choo does. An adversary can amplify this, leading to a kind of DoS attack. Thus, Rai-Choo outperforms PI-Cut-Choo immediately in terms of communication.

Implementation. To demonstrate computational practicality, we have implemented a prototype of our scheme in C++ using above parameter settings. Our implementation uses the MCL library[13] and can be found at

https://github.com/b-wagn/Raichoo

Although our scheme is highly parallelizable, we did not implement any parallelization. To evaluate the efficiency of our implementation, we determined the average running time over 100 runs of the signing interaction (i.e., running U_1, then S, then U_2), and the verification algorithm. For our tests, we used a Intel Core i5-7200U processor @2,5 GHz with 4 cores and 8 GB of RAM, running Ubuntu 20.04.4 LTS 64-bit. Our results are presented in Table 3.3. In general, the table shows a tradeoff between signature size, communication complexity, and computational efficiency.

Other Pairing Settings. We could also instantiate our scheme in the type-2 or type-1 pairing setting. To recall, in the type-2 setting, there is an efficient isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. In the type-1 setting, we have $\mathbb{G}_1 = \mathbb{G}_2$. In both cases, the public keys pk_i in our scheme only have to be sent in $\mathbb{G}_2$. Type-1 pairing-friendly curves are usually constructed from supersingular curves over a field with small characteristics (2 or 3). They were shown to be insecure [BGJT14]. The alternative are supersingular curves over a larger field, but we only know how to construct them with an embedding degree 3. Assuming the same target group size as for BLS12-381 (4572 bits), we get impractical group element sizes of 1525 bits. For type-2 pairings, we can start with the BLS12-381 parameters, and replace the curve $\mathbb{G}_2$ with a curve $\mathbb{G}_2'$ defined over a larger extension field [CM09]. The most efficient way to represent elements in $\mathbb{G}_2'$ is to represent them as an element of $\mathbb{G}_1 \times \mathbb{G}_2$. For more details, see [CM09]. Therefore, by sending all keys in $\mathbb{G}_2'$, we obtain the same communication and signature sizes as in the type-3 setting. The security of such a variant of our scheme will solely rely on the computational Diffie-Hellman assumption in the group $\mathbb{G}_2'$.

Concrete Bit Security. In contrast to what we did in Sections 3.4.2 and 3.5.4, we compute our parameters using standardized curves and hash functions instead of estimating parameters based on the security loss. The reason for this is that we want our numbers be consistent with our implementation and therefore have to rely on standardized components. To discuss the effect of the security loss, we now assume all components are roughly 128-bit secure. Then, the guaranteed security for our scheme is roughly $128 - \log \ell = 98$ bit. This is the same for PI-Cut-Choo and the standard BLS signature scheme [BLS01].

[12] The Python script can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.
[13] See https://github.com/herumi/mcl.

4

Multi-Signatures

Publication History of This Chapter

This chapter is based on the publication [PW23a] and its full version [PW23b] and the publication [PW24] and its full version [PW23c]. I am the main author of both. To obtain a consistent structure, the content of the original publications has been reordered and partially merged. Further, a minor optimization (see Section 4.6), which was not present in [PW23a] but in [PW24], has been applied to the schemes from [PW23a] for the efficiency comparison. Additionally, minor notational changes have been made.

4.1 Introduction

A multi-signature scheme [IN83, BN06] allows a group of signers to jointly sign a message. Naively, every signer could sign the message locally, and we concatenate the resulting signatures. As the number of signers grows large, this results in impractical signature sizes, and so we aim for a more clever solution with compact signatures, potentially at the cost of introducing interaction. Early constructions of multi-signatures have been presented and analyzed in a variety of models [MOR01, Bol03, LOS+06, DEF+19, CKM21], mostly differing in how keys are generated, registered, and verified. Nowadays, the accepted de facto standard for multi-signatures is the plain public key model [BN06], where each signer generates his key pair independently. In this chapter, we focus on constructions in the said model, proven in the random oracle model [BR93] from assumptions over cyclic groups without pairings. We look at this problem from the perspective of concrete security. As explained in Section 1.3, this means we try to optimize the security bound without compromising efficiency.

Three-Round Schemes. In the plain public key model, Bellare and Neven [BN06] have constructed a three-round multi-signature scheme (BN) based on DLOG. Proving the security of this scheme relies on rewinding and uses the Forking Lemma [BN06], which leads to a highly non-tight security bound. To improve this, Bellare and Neven have also introduced a second three-round construction (BN+) tightly based on DDH. Further works focus on key aggregation [MPSW19, BDN18, FH21]. This feature allows to publicly compute a single aggregated key from a given list of public keys, which can later be used for verification. The key aggregation property saves bandwidth and is desirable in many applications. Notably, the three-round scheme Musig [MPSW19, BDN18] can be seen as a variant of BN that supports key aggregation. The scheme is based on DLOG and a double forking technique is introduced for its analysis. This leads to a security bound which is meaningless in terms of concrete security. Using DDH, a tightly secure variant Musig+ of Musig has been proposed in [FH21].

Two-Round Schemes. To further reduce round complexity, recent works have proposed two-round constructions [NRS21, BD21, AB21, CKM21, DOTT21]. However, while achieving certain desirable properties (e.g., deterministic signing [NRSW20]) the proposed schemes have their drawbacks in terms of assumptions and concrete security. The scheme [NRSW20] makes use of heavy cryptographic machinery and is not comparable with others in terms of efficiency. Further, even using additional idealizations such as the algebraic group model, security proofs of most two-round schemes rely on non-standard interactive assumptions [NRS21, CKM21, AB21], the only exceptions being [DOTT21, BD21, BTT22]. A second drawback is the apparent need for (double) forking in the random oracle model [DEF+19, NRS21, DOTT21, BD21, BTT22]. While such security proofs show the absence of major structural attacks, concrete parameters are not supported by cryptanalytic evidence.

Our Goal. Motivated by the state-of-the-art, we study whether rewinding techniques are necessary for two-round multi-signatures. We want to rely on a well-studied non-interactive hardness assumption and construct a scheme without rewinding, ideally with a tight security proof. In spirit of our overall objectives (see Section 1.3), we want to achieve this without sacrificing efficiency.

4.1.1 Contribution: Chopsticks

As our first contribution in this chapter, we propose the first two multi-signature schemes that are two-round from a non-interactive assumption without using the Forking Lemma. Both of our schemes are proven secure in the random oracle model based on the DDH assumption:

1. Chopsticks I. A two-round multi-signature scheme with a security loss $O(Q_S)$ and key aggregation, where Q_S is the number of signing queries.

2. Chopsticks II. The first two-round multi-signature scheme with a fully tight security proof.

We compare our schemes with existing schemes in Table 4.1[1]. For roughly 128-bit security, Chopsticks II can be instantiated with standardized 128-bit secure curves, in contrast to all previous two-round schemes. For Chopsticks I, its proof is non-tight, but it does not rely on rewinding and has tighter security based on standard, non-interactive assumptions than other non-tight schemes (such as HBMS and Musig2). Hence, as long as the number of signing queries Q_S is less than $2^{192-128} = 2^{64}$, we can implement our first scheme with a standardized 192-bit secure curve to achieve 128-bit security, while this is not the case for HBMS and Musig2. As limitations, we note that our schemes do not have some additional beneficial properties (e.g., having Schnorr-compatible signatures or supporting preprocessing) as in Musig2 [NRS21]. We leave achieving these properties without rewinding as an interesting open problem.

A central building block of our construction is a special DDH-based commitment scheme without pairings. Concretely, our commitment scheme has the following properties:

- It commits to pairs of group elements in a homomorphic way.
- It has a dual-mode property, i.e., indistinguishable commitment keys in statistically hiding and statistically binding mode, with tight multi-key indistinguishability.
- The hiding mode offers a special form of equivocation trapdoor, which allows to open commitments to group elements output by the Honest-Verifier Zero-Knowledge (HVZK) simulator of Schnorr-like identification protocols.

Such a commitment scheme can be useful to construct other interactive signature variants, and we believe that this is of independent interest. In this chapter, we construct the first commitment scheme satisfying the above properties simultaneously without using pairings. Our commitment scheme can be seen as an extension of the commitment scheme in [BCJ08][2]. Contrary to our scheme, the commitment scheme in [BCJ08] commits to single group elements and no statistically binding mode is shown. Other previous commitment schemes either have no trapdoor property [GOS06, GS08], or homomorphically commit to ring or field elements [GQ88, Ped92]. To the best of our knowledge, there is only a solution using pairings [Gro09]. Combining this commitment scheme, lossy identification [KW03, AFLT12, KMP16], and a guessing argument, we obtain Chopsticks I. We then introduce the pseudorandom matching technique, where each signer holds two secret keys and pseudorandomly matches its keys to two possible commitment keys per message. Implemented carefully, this idea leads to our tight scheme Chopsticks II.

4.1.2 Contribution: Toothpicks

For our constructions in Chopsticks, the price of tightness is high: signatures and communication complexity in Chopsticks II are about 3 times as large as in one of the most efficient non-tight two-round scheme HBMS. As a second contribution of this chapter, we reduce this efficiency penalty by constructing a new two-round multi-signature scheme (Toothpicks II) that achieves the best of two worlds:

- *Tightness.* Our scheme is tightly secure based on the DDH assumption. When instantiated over a standardized 128-bit secure group, its security guarantee is 126-bit, which is formally supported by our proofs. In contrast, non-tight schemes relying on rewinding do not guarantee any meaningful security level.

[1] We do not consider proofs in the (idealized) algebraic group model and do not list schemes that are not in the plain public key model. Also, scheme TSSHO [TSS+23] has been proposed after our schemes have been published [PW23a], and does not achieve tight security.

[2] Drijvers et al. [DEF+19] showed a flaw in the proof of the multi-signature scheme presented in [BCJ08], but it does not affect their commitment scheme.

Scheme	Rounds	Key Aggregation	Assumption	Loss
BN [BN06]	3	✗	DLOG	$\Theta(Q_H/\epsilon)$
BN+ [BN06]	3	✗	DDH	$\Theta(1)$
Musig [MPSW19, BDN18]	3	✓	DLOG	$\Theta(Q_H^3/\epsilon^3)$
Musig+ [FH21]	3	✓	DDH	$\Theta(1)$
Musig2 [NRS21]	2	✓	AOMDL	$\Theta(Q_H^3/\epsilon^3)$
HBMS [BD21]	2	✓	DLOG	$\Theta(Q_S^4 Q_H^3/\epsilon^3)$
TZ [TZ23]	2	✓	DLOG	$\Theta(Q_H^3/\epsilon^3)$
TSSHO [TSS+23]	2	✓	DDH	$\Theta(Q_S)$
Chopsticks I	2	✓	DDH	$\Theta(Q_S)$
Chopsticks II	2	✗	DDH	$\Theta(1)$
Toothpicks I	2	✓	DDH	$\Theta(Q_S)$
Toothpicks II	2	✗	DDH	$\Theta(1)$

Table 4.1: Comparison of multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. We compare the number of rounds, whether the schemes support key aggregation, the assumption the schemes rely on, and the security loss, where Q_H, Q_S denote the number of random oracle and signing queries, respectively, and ϵ denotes the advantage of an adversary against the scheme. For the security loss, we do not consider proofs in the algebraic group model. We do not list [NRSW20] as it is prohibitively inefficient due to the use of heavy cryptographic machinery.

- *Efficiency.* Our scheme is as efficient as the state-of-the-art non-tight schemes. Concretely, the communication complexity per signer for our scheme is comparable to HBMS [BD21] and about 1.5 times smaller than TZ [TZ23] and Musig2 [NRS21]. The signature size is only about 1.5 times larger than for the non-tight schemes. Compared to Chopsticks II, this significantly reduces the efficiency cost of tightness. Concretely, our scheme outperforms Chopsticks II by a factor of more than 2 in terms of signature size and communication complexity, respectively.

In addition, our techniques allow us to improve the efficiency of Chopsticks I, the non-tight scheme with key aggregation, resulting in Toothpicks I. We compare our schemes with previous schemes in terms of security (see Table 4.1) and asymptotic (see Table 4.2) and concrete (see Table 4.3) efficiency.

From a technical perspective, our first contribution is a new pseudorandom path technique, as opposed to the pseudorandom matching technique used in Chopsticks II. This new technique allows us to reduce the size of signatures and communication by a factor of two. Our second technical insight is that, somewhat surprisingly, we do not need a binding commitment. Instead, we can significantly relax the binding property of our commitment to hold up to cosets of a certain subspace. This enables more efficient instantiations, further improving the efficiency of our schemes. To show that this relaxation does not introduce problems in terms of security, we identify a strong soundness property many natural lossy identification schemes [KW03, AFLT12, KMP16] have.

4.1.3 More on Related Work

Here, we present more related work in the context of multi-signatures. This should serve as a starting point for further reading rather than as an exhaustive overview.

Other Algebraic Structures. In this dissertation, we only consider multi-signatures over pairing-free cyclic groups. Multi-signatures have also been considered over different algebraic structures and using different hardness assumptions. Examples include multi-signatures from RSA [IN83, OO93, HW18], from pairings [BGLS03, Bol03, LOS+06, BDN18, DGNW20], or from lattice-based assumptions [DOTT21, BTT22, FSZ22, Che23, FHSZ23].

Scheme	Rounds	Public Key	Communication	Signature
BN [BN06]	3	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 2\lambda$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
BN+ [BN06]	3	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 2\lambda$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
Musig [MPSW19, BDN18]	3	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 2\lambda$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
Musig+ [FH21]	3	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + 2\lambda$	$2\langle\mathbb{Z}_p\rangle$
Musig2 [NRS21]	2	$1\langle\mathbb{G}\rangle$	$4\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle$
HBMS [BD21]	2	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
TZ [TZ23]	2	$1\langle\mathbb{G}\rangle$	$4\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$1\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$
TSSHO [TSS+23]	2	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle$	$3\langle\mathbb{Z}_p\rangle$
Chopsticks I	2	$2\langle\mathbb{G}\rangle$	$3\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda$	$4\langle\mathbb{Z}_p\rangle + 2\lambda$
Chopsticks II	2	$4\langle\mathbb{G}\rangle$	$6\langle\mathbb{G}\rangle + 2\langle\mathbb{Z}_p\rangle + \lambda + 1$	$8\langle\mathbb{Z}_p\rangle + 4\lambda + N$
Toothpicks I	2	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda$	$3\langle\mathbb{Z}_p\rangle + 2\lambda$
Toothpicks II	2	$4\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1\langle\mathbb{Z}_p\rangle + \lambda + 1$	$3\langle\mathbb{Z}_p\rangle + 2\lambda + N$

Table 4.2: Asymptotic efficiency comparison of multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. We compare the number of rounds, the size of public keys, the communication complexity per signer, and the signature size. We denote the size of a group element by $\langle\mathbb{G}\rangle$ and the size of a field element by $\langle\mathbb{Z}_p\rangle$. Here, λ is a statistical security parameter, and N is the number of signers. Schemes below the line have two rounds and avoid rewinding, see Table 4.1. We do not list [NRSW20] as it is prohibitively inefficient due to the use of heavy cryptographic machinery.

Scheme	Security	Public Key	Communication	Signature
Musig2	9	33	164	65
HBMS	-11	33	97	97
TZ	8	33	196	97
TSSHO	106	66	130	96
Chopsticks I	106	66	147	160
Chopsticks II	126	132	278	336
Toothpicks I	106	66	114	128
Toothpicks II	125	132	114	144

Table 4.3: Concrete efficiency and security comparison of two-round multi-signature schemes in the discrete logarithm setting without pairings in the plain public key model. We compare the security level guaranteed by the security bound in the random oracle model assuming the underlying assumption is 128-bit hard, the size of public keys, the communication complexity per signer, and the signature size. Sizes are given in bytes and rounded. We do not list [NRSW20] as it is prohibitively inefficient due to the use of heavy cryptographic machinery.

Aggregate Signatures. Aggregate signatures [BGLS03, LMRS04, LOS+06, HKW15] allow to publicly aggregate a set of given signatures. More precisely, given the public keys, messages, and signatures, one can compute one short aggregate signature that represents all signatures. In this sense, we can think of aggregate signatures as non-interactive multi-signatures for different messages. In fact, one caveat of most aggregate signatures is that the messages necessarily have to be distinct.

Synchronized Multi-Signatures. Multi-signatures and aggregate signatures in the synchronized setting have been considered [AGH10, HW18, FSZ22, FHSZ23]. In a nutshell, these constructions assume that each signer only signs one message per time period and only signatures from the same time period can be aggregated. This assumption is valid in several applications and enables more efficient and non-interactive constructions in various settings.

Threshold Signatures. Threshold signatures [Des88, DF90, Ped91] generalize multi-signatures in the sense that not all signers have to participate to generate a signature. Precisely, in a threshold signature scheme, a group of n signers is represented by one public key, and for a threshold t, a valid signature with respect to this public key can only be computed when more than t signers participate. It is worth highlighting the difference between multi-signatures and threshold signatures with threshold $t = n - 1$: in the multi-signature setting, we assume all signers generate their key independently and aggregation of public keys may be possible, whereas in the threshold signature setting, we assume that the group of n signers is fixed and secret keys can be generated in a correlated way. We discuss threshold signatures extensively in Chapter 5.

4.1.4 Subsequent and Concurrent Work

In a concurrent work to our Chopsticks result [PW23a], Tessaro and Zhu [TZ23] also proposed a two-round multi-signature scheme, among other contributions. Both our work and theirs focus on avoiding interactive assumptions. However, while we additionally remove the security loss, Tessaro and Zhu concentrate on having a partially non-interactive scheme. That is, the first round of the signing protocol is independent of the message being signed. In a nutshell, they generalize Musig2 to linear function families. Then, under a suitable instantiation, the interactive assumption for Musig2 can be avoided. Similar to Musig2, the resulting scheme is partially non-interactive. Subsequent to our Chopsticks result [PW23a], but before our Toothpicks result [PW24], Takemure et al. [TSS+23] proposed a two-round multi-signature scheme from DDH similar to our non-tight scheme Chopsticks I: it supports key aggregation and has a security loss proportional to the number of signing queries. Roughly, they combine the Katz-Wang signature scheme [KW03] with techniques inspired by HBMS [BD21]. As shown in Table 4.3 their scheme is particularly efficient in terms of signature size.

4.1.5 Outline

Following the introduction of this chapter, we give a detailed technical overview in Section 4.2, which presents our techniques in an informal way. Then, in Section 4.3, we define two-round multi-signatures. The remaining two sections present our contributions formally. That is, Section 4.4 presents our Chopsticks schemes from [PW23a], and Section 4.5 presents our Toothpick schemes from [PW24], for which we also discuss concrete parameters and efficiency.

4.2 Technical Overview

In this overview, we informally introduce the main ideas and techniques developed in this chapter. We start with our Chopsticks construction from [PW23a], and then explain our Toothpicks construction from [PW24]. To recall, the goal of Chopsticks is to develop techniques for constructing two-round multi-signature schemes without using rewinding. The goal of Toothpicks is to reduce the efficiency penalty introduced by these techniques.

4.2.1 Fork-Free Two-Round Multi-Signatures

Our first contribution is to construct two-round multi-signatures without using rewinding. Here, we present our techniques informally.

Schnorr-Based Multi-Signatures. We start by recalling the basic template for multi-signatures based on the Schnorr identification scheme [Sch91]. Let $\mathbb{G}$ be a group of prime order p with generator g. We explain the template using the vector space homomorphism $\mathsf{F}\colon x \mapsto g^x$ mapping from $\mathbb{Z}_p$ to $\mathbb{G}$, and write both domain and range additively. In a first approach to get a multi-signature scheme, we let each signer i with secret key sk_i and public key $\mathsf{pk}_i = \mathsf{F}(\mathsf{sk}_i)$ sample a random $r_i \in \mathbb{Z}_p$, and send $R_i := \mathsf{F}(r_i)$ to all other signers. Then, an aggregated R is computed as $R = \sum_i R_i$. From this R, signers derive challenges c_i using a random oracle. Then, each signer computes a response $s_i = c_i\mathsf{sk}_i + r_i$ and sends this response. Finally, the signature contains R and the aggregated response $s = \sum_i s_i$. Verification is very similar to the verification of Schnorr signatures. As each signer in this simple two-round scheme is almost identical to the prover algorithm of the Schnorr identification scheme, one may hope that this scheme is secure. However, early works already noted that it is not [BN06].

While there are concrete attacks against the scheme, for our purposes it is more important to understand where the security proof fails. The proof fails when we try to simulate honest signer without knowing its secret key sk_1. Following Schnorr signatures and identification, this would be done by sampling $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random c_1 an s_1, and then programming the random oracle accordingly at position R. The problem in the multi-signature setting is that we first have to output R_1, and then the adversary can output the remaining R_i, such that he has full control over the aggregate R. Thus, the random oracle may already be defined. Previous works [BN06, MPSW19, BDN18] solve this issue by introducing an additional round in which all signers commit to their R_i using a random oracle. This allows us to extract all R_i from these commitments in the reduction, and therefore R has enough entropy to program the random oracle.

A second problem that we encounter in the above approach is the extraction of a solution from the forgery. Namely, to extract a discrete logarithm of pk_1, we need to rely on rewinding. Some of the well-known schemes [MPSW19, BDN18] even use rewinding multiple times. This leads to security bounds with essentially no useful quantitative guarantee for concrete security.

Towards A Scheme without Rewinding. To avoid rewinding, our first idea is to rely on a different homomorphism F. Namely, we borrow techniques from lossy identification [KW03, AFLT12, KMP16] and use $\mathsf{F}\colon x \mapsto (g^x, h^x)$ for a second generator $h \in \mathbb{G}$. We can then give a non-rewinding security proof for the three-round schemes in [BN06, MPSW19, BDN18]. Concretely, we first switch pk_1 from the range of F to a random element in $\mathbb{G}^2$, using the DDH assumption. Then, we can argue that a forgery is hard to compute using a statistical argument known as *lossy soundness*. We note that this idea is (implicitly) already present in [BN06, FH21]. As we will see, combining it with techniques to avoid the extra round is challenging.

Towards Two-Round Schemes. To go from a three-round scheme as above to a two-round scheme, our goal is to avoid the first round. Recall that this round was needed to simulate R_1 using random oracle programming. Our idea to tackle the simulation problem is a bit different. Namely, going back to the (insecure) two-round scheme, our goal is to send R_1 *after* we learn c_1. If we manage to do that, we can simulate by setting it as $R_1 := \mathsf{F}(s_1) - c_1\mathsf{pk}_1$ for random s_1. Of course, just sending R_1 after learning c_1 should only be possible for the reduction. Following Damgård [Dam00], this high-level strategy can be implemented using a trapdoor commitment scheme Com, and sending $\mathsf{com}_1 = \mathsf{Com}(\mathsf{ck}, R_1)$ as the first message instead of R_1. The challenges c_i are then derived from an aggregated commitment com using the random oracle. Later, the reduction can open this commitment to $\mathsf{F}(s_1) - c_1\mathsf{pk}_1$ using the trapdoor for commitment key ck. To support aggregation, the commitment scheme should have homomorphic properties. Note that this approach has been used in the lattice setting in a recent work [DOTT21]. However, implementing such a commitment scheme for (pairs of) group elements is highly non-trivial, as we will see. Also, as already pointed out in [DOTT21], it is hard to make this two-round approach work while avoiding rewinding at the same time. The reason is

that a trapdoor commitment scheme can not be statistically binding. But if we want to make use of lossy soundness discussed above, we need that R is fixed before the c_i are sampled, which requires statistical binding. With a computationally binding commitment scheme, we end up in a rewinding reduction (to binding) again. Our first technique will help us to overcome this issue.

Chopstick One: Our Scheme Without Rewinding. Our idea to overcome the above problem is to demand a dual-mode property from the commitment scheme Com. Namely, there should be an indistinguishable second way to set up the commitment key ck, such that for such a key the scheme is statistically binding. This does not solve the problem yet, because we require ck to be in trapdoor mode for simulation, and in binding mode for the final forgery. The solution is to sample ck in a message-dependent way using another random oracle, which is (for other reasons) already done in earlier works [DEF+19, DOTT21]. In this way, we can embed a binding commitment key in some randomly guessed random oracle queries, and a trapdoor key in others. Note that this requires a tight multi-key indistinguishability of the commitment scheme. Assuming we have such a commitment scheme, we end up with our first construction $\mathsf{Chopsticks\ I}$. Of course, this strategy still has a security loss linear in the number of signing queries due to the guessing argument, but it avoids rewinding. In addition, we can implement the approach in a way that supports key aggregation.

Chopstick Two: Our Fully Tight Scheme. The security loss in our first scheme results from partitioning random oracle queries into two classes, namely queries returning binding keys, and queries returning trapdoor keys. To do such a partitioning in a tight way, we may try to use a Katz-Wang random bit approach [GJKW07]. This simple approach can be used for standard digital signatures. However, it turns out that it does not work for our case. To see this, recall that naively following this approach, we would compute two message-dependent commitment keys

$$\mathsf{ck}_0 := \mathsf{H}(0, \mathsf{m}), \quad \mathsf{ck}_1 := \mathsf{H}(1, \mathsf{m}).$$

Then, for each message, we would embed a binding key in one branch, and a trapdoor key in the other branch, e.g., ck_0 binding and ck_1 with trapdoor. In the signing protocol, we would abort one of the branches pseudorandomly based on the message. Then we could use the trapdoor branch in the signing, and hope that the forgery uses the binding branch. However, this strategy crucially relies on the fact that the aborting happens in a way that is pseudorandom to the adversary. Otherwise the adversary could always choose the trapdoor branch for his forgery. While we can implement this in a signature scheme, in our multi-signature scheme this fails, because all signers must use the *same commitment key* to make aggregation possible. At the same time, the aborted branch must depend on secret data of the simulated signer to remain pseudorandom.

To solve this problem, we observe that the above approach uses a pseudorandom "branch selection" and aborts the other branch. Our solution can be phrased as a pseudorandom "branch-to-key matching". Namely, we give each signer two public keys $(\mathsf{pk}_{i,0}, \mathsf{pk}_{i,1})$. The signing protocol is run in two instances in parallel. One instance uses ck_0, and one uses ck_1 as above. More precisely, we commit to R_0 via ck_0 and to R_1 via ck_1. Then we aggregate and determine the challenges $c_{i,0}$ and $c_{i,1}$. However, before sending the response $s_i = (s_{i,0}, s_{i,1})$, *each signer separately* determines which key to use in which instance, i.e., it computes

$$s_{i,0} = c_{i,0} \cdot x_{i,b_i} + r_{i,0}, \quad s_{i,1} = c_{i,1} \cdot x_{i,1-b_i} + r_{i,1},$$

where b_i is a pseudorandom bit that each signer i computes *independently*, and that will be included in the final signature to make verification possible. This decouples the public key that is used from the commitment key that is used. Now, we are ready to discuss the implication of this change. Namely, our reduction chooses $\mathsf{pk}_{1,0}$ honestly and $\mathsf{pk}_{1,1}$ as a lossy key, i.e., random instead of in the range of F. Then, in each signing interaction, the reduction can match the honest public key with the binding commitment key and the lossy public key with the trapdoor commitment key by setting b_1 accordingly. In this way, we can simulate one branch using the actual secret key, and the other branch using the commitment trapdoor. For the forgery, we hope that the matching is the other way around, such that

binding commitment key and lossy public key match, which makes the statistical argument from lossy identification possible. Overall, this approach leads to our fully tight scheme Chopsticks II.

The Challenge of Instantiating the Commitment. One may observe that we shifted a lot of the challenges that we encountered into properties of the underlying commitment scheme. This naturally raises the question if such a commitment scheme can be found. In fact, constructing this commitment scheme can be understood as our second technical main contribution.

Let us first explain why it is non-trivial to construct such a scheme. The main barrier results from the algebraic structure that we demand. Namely, we need to commit to group elements[3] $R \in \mathbb{G}$. A naive idea would be to use any trapdoor commitment scheme, e.g., Pedersen commitments, by first encoding R in the appropriate message space. However, this would destroy all homomorphic properties that we need, and we should not forget that we need a dual-mode property. This brings us to Groth-Sahai commitments [GS08], which can commit to group elements. Indeed, these commitments are homomorphic, and have (indistinguishable from) random keys, such that we can sample them using a random oracle. They are also dual-mode based on DDH, which allows us to use the random self-reducibility of DDH to show tight multi-key indistinguishability. However, the trapdoor property turns out to be the main challenge. To see why this is problematic, note that the opening information of these commitments typically contains elements from $\mathbb{Z}_p$ that are somehow used as exponents. There are exceptions to this rule, like [Gro09], but they use pairings and the DLIN assumption, which we aim to avoid. This means that the trapdoor should allow us to sample exponents, given a group element R to which we want to open the commitment. This naturally corresponds to having a trapdoor for the discrete logarithm problem, which we do not have.

Our Solution: Weakly Equivocable Commitments. Our starting point is the commitment scheme for group elements given in [GS08]. Namely, commitment keys correspond to matrices $\mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{2\times 2}$, and to commit to a message $R = g^r \in \mathbb{G}$ with randomness $(\alpha, \beta) \in \mathbb{Z}_p$, one computes

$$\mathsf{com} := (C_0, C_1)^t := \left(A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta}, R \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta}\right)^t.$$

That is, setting $\mathbf{E} = (E_{i,j})_{i,j} \in \mathbb{Z}_p$ such that $g^{E_{i,j}} = A_{i,j}$, we can write the discrete logarithm of com as $(0, r)^t + \mathbf{E} \cdot (\alpha, \beta)^t$. In binding mode, matrix $\mathbf{E}$ is a matrix of rank 1, while $\mathbf{E}$ has full rank in hiding mode. It is easy to see that this commitment scheme to group elements is homomorphic. However, we stress that there is no simple solution to implement a trapdoor for equivocation. To see this, note that if we want to open a commitment com to a message $R' \in \mathbb{G}$, we need to output a suitable tuple (α, β). If we knew the discrete logarithm of com, then we still would need to know the discrete logarithm of R' to find such a tuple. The key insight of our trapdoor construction is that we do not need to be able to open com to any message R'. Instead, it will be sufficient if we can open it to messages of the form $R' = g^s \cdot \mathsf{pk}^c$, where we do not know c when we fix the commitment com, but *we know* pk *when setting up* $\mathbf{A}$. To explain why this helps, assume we want to find a valid opening (α, β) in this case. Then we need to satisfy

$$\mathsf{com} = \begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} 0 \\ g^s \mathsf{pk}^c \end{pmatrix} \cdot g^{\mathbf{E}\cdot(\alpha,\beta)^t}.$$

It seems like we did not make progress, because even if we know the discrete logarithms of C_0, C_1, the term pk^c is not known in the exponent. Now, our key idea to solve this is to write and generate $\mathbf{A}$ with respect to basis pk in the second row. Namely, we generate $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} \\ \mathsf{pk}^{d_{2,1}} & \mathsf{pk}^{d_{2,2}} \end{pmatrix}.$$

[3] In the actual construction, we need to commit to pairs of group elements, but we consider the simpler setting of one group element in this overview.

In this way, the equation that we need to satisfy becomes

$$\begin{pmatrix} C_0 \\ C_1 \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha+d_{1,2}\beta} \\ g^s \mathsf{pk}^{c+d_{2,1}\alpha+d_{2,2}\beta} \end{pmatrix}.$$

Next, we get rid of the term g^s by shifting C_1 accordingly. Namely, recall that we can sample s at random long before we learn c. Setting $C_0 = g^\tau$ and $C_1 = g^s \mathsf{pk}^\rho$ for random τ, ρ, we obtain the equation

$$\begin{pmatrix} g^\tau \\ \mathsf{pk}^\rho \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha+d_{1,2}\beta} \\ \mathsf{pk}^{c+d_{2,1}\alpha+d_{2,2}\beta} \end{pmatrix}.$$

Given the trapdoor $\mathbf{D} = (d_{i,j})_{i,j}$, this can easily be solved for (α, β) by solving $(\tau, \rho - c)^t = \mathbf{D} \cdot (\alpha, \beta)^t$. We are confident that such a weak and structured equivocation property can be used in other applications as well.

4.2.2 Reducing the Price of Tightness

The techniques discussed above lead to an efficiency overhead that we want to avoid. In the second part of this technical overview, we present our main ideas to do so. To help the reader, some ideas used in the Chopsticks schemes that we already discussed above are recalled along the way.

Lossy Identification and Commitments. To recall, the Chopsticks schemes use lossy identification schemes [KW03, AFLT12, KMP16]. In such an identification scheme, there are two ways to set up public keys pk. As usual, one can set up pk with a secret key sk. In the specific setting we consider, this means $\mathsf{pk} = \mathsf{F}(\mathsf{sk})$. Alternatively, one can set up pk in lossy mode. In this mode, not even an unbounded prover can make the verifier accept, which is called lossy soundness. To make use of this primitive in the two-round multi-signature setting, recall that we have constructed and leveraged a homomorphic dual-mode commitment. Concretely, such a commitment has two ways of setting up commitment keys ck. In the hiding mode, ck is generated in combination with a (weak) equivocation trapdoor. In the other mode, commitments are statistically binding. Given such a commitment and a lossy identification scheme, a Chopsticks signature for a message m contains transcripts of the lossy identification scheme, where some parts are given in a committed form. For these parts, the signature also contains the respective opening information. Importantly, the commitment key is derived from m, e.g., as $\mathsf{ck} := \mathsf{H}(\mathsf{m})$, where H is a random oracle. Abstractly, we have identified the following properties:

- *Simulation via Secret Keys.* A reduction can simulate the signing oracle using the secret key if pk is in the normal mode. The mode of ck is not relevant.
- *Simulation via Trapdoors.* A reduction can simulate the signing oracle using the trapdoor if ck is in the hiding mode. The mode of pk is not relevant.
- *Forgery.* To show security without rewinding, the adversary must output a forgery with respect to a lossy pk and a binding ck.

The Chopsticks Approach: Pseudorandom Matching. In the proof of Chopsticks I, we use these properties by sampling all commitment keys with a trapdoor, allowing us to simulate signing even if the public key is lossy. Only for the forgery message the associated commitment key ck^* is set up to be binding. Then, the proof can be finished without rewinding. On the downside, this approach requires guessing the query defining ck^*, leading to a security loss.

A well-known trick to avoid such a guessing argument is the Katz-Wang approach [GJKW07]. Here, each message would specify two commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \mathsf{m})$ and $\mathsf{ck}_1 := \mathsf{H}(1, \mathsf{m})$, and a signer individually would pick a pseudorandom bit b_m for each message and then use $\mathsf{ck}_{b_\mathsf{m}}$. As we have seen, it turns out that this trick is not applicable here, as each signer has to use the same commitment key.

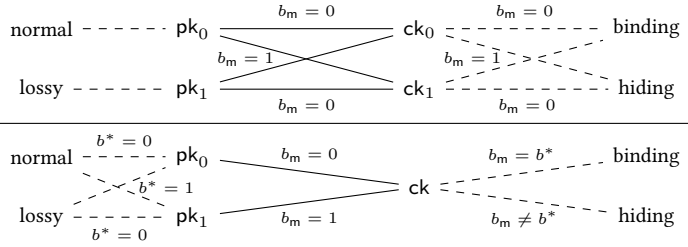

Figure 4.1: Visualization of the pseudorandom matching technique from Chopsticks [PW23a] (top), and our new pseudorandom path technique (bottom). Here, b^* is a random bit sampled by the game, b_m is the pseudorandom bit that the signer chooses for message m, normal edges indicate how keys are paired in the scheme, and dotted edges indicate how keys are set up in the proof.

To overcome this obstacle and construct Chopsticks II, recall that we have proposed the pseudorandom matching technique. Namely, each signer has two public keys $\mathsf{pk}_0, \mathsf{pk}_1$, and both message-dependent commitment keys ck_0 and ck_1 are used. That is, the protocol is run twice in parallel and the signature now contains two transcripts instead of one. Importantly, each signer uses the pseudorandom bit b_m to decide which public key to match with which commitment key. We illustrate the pseudorandom matching technique in Figure 4.1 (top). In the proof of Chopsticks II, one can set pk_1 to lossy and always match it with the trapdoor commitment key $\mathsf{ck}_{1-b_\mathsf{m}}$ for signing queries. In this way, it is possible to simulate the $(\mathsf{pk}_1, \mathsf{ck}_{1-b_\mathsf{m}})$ side via the trapdoor, and the $(\mathsf{pk}_0, \mathsf{ck}_{b_\mathsf{m}})$ side via the secret key. At the same time, with probability $1/2$, the adversary will match the lossy pk_1 with the binding $\mathsf{ck}_{b_\mathsf{m}}$ for the forgery, finishing the proof. While this trick works well, it introduces a significant overhead for both signature size and communication complexity.

Our New Approach: Pseudorandom Paths. We avoid this overhead by using our new pseudorandom path technique, as illustrated in Figure 4.1 (bottom). Our first observation is that for the argument used to finish the proof of Chopsticks II, only one of the two paths, namely the $(\mathsf{pk}_1, \mathsf{ck}_{b_\mathsf{m}})$ path, is used. Instead of simply omitting one of the paths, which leads to problems similar to the naive Katz-Wang approach, let us see what happens if we go back to a solution in which there is only one commitment key ck per message m. If we also reduce the number of keys per signer back to one, we end up with the guessing-based solution again. So, we keep the two keys $\mathsf{pk}_0, \mathsf{pk}_1$ per signer, and let each signer pseudorandomly decide which key $\mathsf{pk}_{b_\mathsf{m}}$ to use in the signing interaction. In our proof, we can set up ck with a trapdoor if the lossy key pk_1 is used, and we can set it up in binding mode if the normal key pk_0 is used. Unfortunately, without additional tricks, this strategy is doomed: the adversary could always use pk_0 in its forgery, which is not lossy. In our final solution, we therefore pick a bit b^* at random at the beginning of our simulation. Then, we set pk_{b^*} to normal and pk_{1-b^*} to lossy. We adapt the sampling of ck accordingly. By carrying out all arguments in the correct order, we can argue that in one of four cases, the adversary used the lossy key pk_{1-b^*} with a binding commitment key in its forgery. Let us explain the idea for that with our illustration (Figure 4.1, bottom) at hand. Every signature corresponds to a pseudorandom path from the left to the right. The bits are set up in a way that ensures the following:

- *Simulation of Signing.* If $\mathsf{pk}_{b_\mathsf{m}}$ is used, the path connects the lossy vertex to the trapdoor vertex, or the normal vertex to the binding vertex. In both cases, we can simulate.
- *Forgery.* The probability that the path associated with the forgery starts at the lossy vertex is $1/2$ and conditioned on that, the probability that the path ends at the binding vertex is also $1/2$.

With this technique, communication and signatures now only consist of one transcript of the lossy identification scheme, as opposed to two transcripts in the pseudorandom matching technique.

The Chopsticks Commitment. So far, we have reduced the size of signatures and communication of

Chopsticks II by a factor of two. At this point, the size is mostly dominated by the size of commitments com and openings φ. It is therefore instructive to recall the commitment instantiation from our Chopsticks schemes and see if we can optimize it. In contrast to our simplified overview we gave before, we now consider the full scheme that allows to commit to pairs of group elements, instead of just a single group element: in Chopsticks I and Chopsticks II, we commit to pairs of group elements $(R_1, R_2) \in \mathbb{G}^2$ via the equation

$$\mathsf{com} = (C_0, C_1, C_2) \in \mathbb{G}^3, \text{ where } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_2 \cdot & A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix}.$$

Here, $\varphi = (\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$ is the commitment randomness and the $A_{i,j} \in \mathbb{G}$ form the commitment key. In terms of exponents, the commitment has the form

$$\mathbf{E} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} + \begin{pmatrix} 0 \\ r_1 \\ r_2 \end{pmatrix}$$

for a matrix $\mathbf{E} \in \mathbb{Z}_p^{3\times 3}$ that determines the commitment key. Now, one can prove that this is statistically hiding if $\mathbf{E}$ has full rank, and it is statistically binding if $\mathbf{E}$ has rank 1. As we have seen (in a simplified form), there is a weak equivocation trapdoor for this commitment, i.e., a trapdoor that allows to open commitments to messages (R_1, R_2) of a certain structure.

Strawman Commitment. In terms of efficiency, note that the 3×3 commitment key leads to three group elements per commitment and three exponents per opening. To improve it, our naive idea is to replace this 3×3 structure with a 2×2 structure, thereby saving one group element and exponent. Concretely, we could try to drop the first row of the commitment equation, leading to

$$\mathbf{E} \cdot \begin{pmatrix} \beta \\ \gamma \end{pmatrix} + \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$$

for a matrix $\mathbf{E} \in \mathbb{Z}_p^{2\times 2}$. Implemented carefully, this is still perfectly hiding with a weak equivocation trapdoor if $\mathbf{E}$ has full rank. Unfortunately, we fail when analyzing the statistically binding mode[4] if $\mathbf{E}$ has rank 1. Concretely, an (unbounded) adversary against binding could output (r_1, r_2) with opening (β, γ) on the one hand, and $(r_1, r_2) + (\beta, \gamma)\mathbf{E}^t$ with opening $(0, 0)$ on the other hand. The first row in the 3×3 scheme prevents this. To save our 2×2 construction without reintroducing such a first row, we thus need additional insights.

Coset Binding. As we have seen, our 2×2 scheme is not (statistically) binding, and as such it is not suitable to instantiate the multi-signature construction. However, we make the crucial observation that the scheme is binding *up to a difference in the span of* $\mathbf{E}$. In other words, if we interpret the commitment as a commitment to cosets of the span of $\mathbf{E}$, the scheme is binding. We call this property *coset binding*. It is instructive to pinpoint where the overall multi-signature proof (of Chopsticks I, Chopsticks II, and our new abstract scheme) fails if we relax binding to coset binding: In the proof of our multi-signature construction, binding shows up in combination with lossy soundness in the very last proof step. To recall, lossy soundness states that even an unbounded prover can not make a verifier of the lossy identification scheme accept, given that pk is in lossy mode. An accepting transcript of the identification scheme has the form (R, c, s) and satisfies $\mathsf{F}(s) - c \cdot \mathsf{pk} = R$ for the linear function F[5].

[4] Recall: if we only have a computationally binding mode, the resulting multi-signature scheme needs to rely on rewinding. Therefore, we have to insist on a statistically binding mode.

[5] To clarify, we use additive notation when talking about lossy identification from such linear functions in general, and multiplicative notation for the concrete instantiation of the linear function and commitment.

Roughly, when constructing an unbounded reduction that breaks lossy soundness, we first guess[6] the random oracle query associated with the forgery. On this query, assume now that the adversary sends a commitment com for R with respect to a binding commitment key. In this case, the reduction would non-efficiently extract the committed pair of group elements $R = (R_1, R_2)$ from com and output it in the lossy soundness game. Further, it would appropriately embed the challenge c from the lossy soundness game. Finally, if the guess was correct, the adversary's forgery contains a valid response s, which the reduction can output. Now, it is clear why coset binding is not enough: the adversary is not bound to $R = (R_1, R_2)$, and it can output a response s that is valid for $R' = (R'_1, R'_2) \neq R$, which is of no use for the reduction.

Coset Lossy Soundness to the Rescue. While coset binding seems to be insufficient at first glance, it still gives us a guarantee we may leverage. Namely, by coset binding, $(R_1 = g^{r_1}, R_2 = g^{r_2})$ and $(R'_1 = g^{r'_1}, R'_2 = g^{r'_2})$ as above have to satisfy that $(r_1, r_2)^t - (r'_1, r'_2)^t$ is in the span of $\mathbf{E}$. We want to understand the impact of this guarantee on the lossy soundness game. For that, imagine a modified lossy soundness game where the final equation $\mathsf{F}(s) - c \cdot \mathsf{pk} = R$ only has to hold up to a difference in the span of $\mathbf{E}$. We call this stronger notion of lossy soundness *coset lossy soundness*. In fact, if we can argue that coset lossy soundness holds, then the reduction sketched above goes through assuming coset binding. For that, our main idea is to set up the binding commitment keys such that the span of $\mathbf{E}$ is always contained in the image of F. In this case, we observe that coset lossy soundness is implied by the original lossy soundness notion. This is because, roughly, if $\mathsf{F}(s) - c \cdot \mathsf{pk}$ equals R up to a difference in the span of $\mathbf{E}$, it means that $\mathsf{F}(s) - c \cdot \mathsf{pk} = R + \mathsf{F}(\delta)$ for some δ, and so one can just treat $s - \delta$ as the new s. To summarize our optimized commitment construction, we have seen that lossy soundness of the identification scheme at hand is strong enough to compensate for the relaxed binding notion. Applying this new commitment to the non-tight Chopsticks I, we obtain Toothpicks I, and in combination with our pseudorandom path technique, we obtain Toothpicks II.

4.3 Preliminaries for this Chapter

In this section, we define multi-signatures, precisely, two-round multi-signatures. We also define key aggregation for two-round multi-signatures.

Two-Round Multi-Signatures. We consider the plain public key model following [BN06]. In this model, each party independently generates a pair of keys $(\mathsf{pk}, \mathsf{sk})$ using a key generation algorithm Gen. Then, signers with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ can come together to sign a message m, using a two-round signing protocol specified by three algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$. In this protocol, each signer takes as input its own secret key, the set of public keys, and the message. Following previous works, e.g., [CKM21, DOTT21], we assume the public keys participating in the signing protocol are given by a set, as opposed to a multi-set or a list. We also assume that sets can be ordered canonically, e.g., lexicographically. With this assumption, we can uniquely encode sets $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, and we denote such an encoding by $\langle \mathcal{P} \rangle$ throughout the chapter. We assume that in each round of the signing protocol (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1$), each signer outputs a message to be sent to all other participating signers. The final algorithm Sig_2 outputs the signature. Signatures can be verified using algorithm Ver with respect to a set of public keys $\mathcal{P}$ and a message m. To summarize a signing interaction in our syntax, we define algorithm $\mathsf{MS.Exec}$ in Figure 4.2.

Definition 4.1 (Multi-Signature Scheme)**.** *A (two-round) multi-signature scheme is a tuple of PPT algorithms* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs global system parameters* par*. We assume that* par *implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to* MS *take* par *at least implicitly as input.*

[6] Recall that lossy soundness is a statistical notion, and so guessing is not a problem in terms of tightness at this point.

Alg $\mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m})$
01 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$, $\{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\} := \mathcal{S}$
02 **for** $i \in [N] : (\mathsf{pm}_{1,i}, St_{1,i}) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
03 $\mathcal{M}_1 := (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$
04 **for** $i \in [N] : (\mathsf{pm}_{2,i}, St_{2,i}) \leftarrow \mathsf{Sig}_1(St_{1,i}, \mathcal{M}_1)$
05 $\mathcal{M}_2 := (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$
06 **for** $i \in [N] : \sigma_i \leftarrow \mathsf{Sig}_2(St_{2,i}, \mathcal{M}_2)$
07 **if** $\exists i \neq j \in [N]$ s.t. $\sigma_i \neq \sigma_j$: **return** $\perp$
08 **return** $\sigma := \sigma_1$

Figure 4.2: Algorithm MS.Exec for a multi-signature scheme $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. The algorithm specifies an honest execution of the signing protocol Sig among N signers with public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_N$ and secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$ for a message m.

- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input system parameters* par*, and outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Sig} = (\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2)$ *is split into three algorithms:*
  - $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m}) \to (\mathsf{pm}_1, St_1)$ *takes as input a set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, a secret key* sk*, and a message* m*, and outputs a protocol message* pm_1 *and a state* St_1*.*
  - $\mathsf{Sig}_1(St_1, \mathcal{M}_1) \to (\mathsf{pm}_2, St_2)$ *takes as input a state* St_1 *and a tuple* $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ *of protocol messages, and outputs a protocol message* pm_2 *and a state* St_2*.*
  - $\mathsf{Sig}_2(St_2, \mathcal{M}_2) \to \sigma_i$ *takes as input a state* St_2 *and a tuple* $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ *of protocol messages, and outputs a signature* σ*.*
- $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input a set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0, 1\}$*.*

We require that MS *is complete in the following sense. For all* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, all* $N = \mathsf{poly}(\lambda)$*, all* $(\mathsf{pk}_j, \mathsf{sk}_j) \in \mathsf{Gen}(\mathsf{par})$ *for* $j \in [N]$*, and all messages* m*, we have*

$$\Pr\left[\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma) = 1 \;\middle|\; \begin{array}{l} \mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}, \mathcal{S} = \{\mathsf{sk}_1, \ldots, \mathsf{sk}_N\}, \\ \sigma \leftarrow \mathsf{MS.Exec}(\mathcal{P}, \mathcal{S}, \mathsf{m}) \end{array}\right] = 1,$$

where algorithm MS.Exec *is defined in Figure 4.2.*

Our security definition is the standard unforgeability notion for multi-signatures and is in line with previous works [BN06, DOTT21]. Namely, we assume that there is one honest signer with an honestly generated public key. This signer is controlled by the game, whereas all other keys and signers are controlled by the adversary. The adversary gets the public key and can interact with this honest signer in signing interactions of its choice. As the adversary controls all other parties, this especially means that we do not assume any broadcast channel. Throughout, we always assume that the honest public key is the entry pk_1 in a set $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ of participating keys. This is without loss of generality and simplifies presentation. Finally, the adversary has to output a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The adversary wins if the honest public key has to be in $\mathcal{P}^*$, σ^* is a valid signature with respect to $\mathcal{P}^*$ and message m^*, and the pair $(\mathcal{P}^*, \mathsf{m}^*)$ is fresh, meaning that the adversary never started a signing interaction for this pair.

Definition 4.2 (MS-EUF-CMA Security)**.** *Let* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a multi-signature scheme and consider the game* $\mathbf{MS\text{-}EUF\text{-}CMA}$ *defined in Figure 4.3. We say that* MS *is* MS-EUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) := \Pr\left[\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda) \Rightarrow 1\right].$$

Game MS-EUF-CMA$^{\mathcal{A}}_{\mathsf{MS}}(\lambda)$

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$
$\textsc{Sig} := (\textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2)$
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sig}}(\mathsf{par}, \mathsf{pk})$
if $\mathsf{pk} \notin \mathcal{P}^*$: **return** 0
if $(\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
return $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Oracle $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$

parse $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
if $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\bot$
$\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\}$
$ctr := ctr + 1,\ sid := ctr$
$\mathsf{round}[sid] := 1$
$(\mathsf{pm}_1, St_1) \leftarrow \mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}, \mathsf{m})$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_1, St_1)$
return $(\mathsf{pm}_1[sid], sid)$

Oracle $\textsc{Sig}_1(sid, \mathcal{M}_1)$

if $\mathsf{round}[sid] \neq 1$: **return** $\bot$
parse $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
if $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
$(\mathsf{pm}_2, St_2) \leftarrow \mathsf{Sig}_1(St_1[sid], \mathcal{M}_1)$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_2, St_2)$
return $\mathsf{pm}_2[sid]$

Oracle $\textsc{Sig}_2(sid, \mathcal{M}_2)$

if $\mathsf{round}[sid] \neq 2$: **return** $\bot$
parse $(\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N}) := \mathcal{M}_2$
if $\mathsf{pm}_2[sid] \neq \mathsf{pm}_{2,1}$: **return** $\bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
$\sigma \leftarrow \mathsf{Sig}_2(St_2[sid], \mathcal{M}_2)$
return σ

Figure 4.3: The game **MS-EUF-CMA** for a (two-round) multi-signature scheme MS and an adversary $\mathcal{A}$. To simplify the presentation, we assume that the canonical ordering of sets is chosen such that pk is always at the first position if it is included.

Key Aggregation. An additional feature that multi-signatures can have is key aggregation. A scheme supports key aggregation if one can compress a set of public keys into one short aggregated public key that represents the set in a sense that signatures are verified with respect to this aggregated public key. One may expect that this also requires to change the security definition. However, as explained in detail in [NRS21] (Appendix C in the eprint version [NRS20]), this is not the case.

Definition 4.3 (Key Aggregation)**.** *A multi-signature scheme* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *is said to support key aggregation, if the algorithm* Ver *can be split into two deterministic polynomial time algorithms* Agg, VerAgg *with the following syntax:*

- $\mathsf{Agg}(\mathcal{P}) \to \tilde{\mathsf{pk}}$ *takes as input a set* $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ *of public keys and outputs an aggregated key* $\tilde{\mathsf{pk}}$.
- $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input an aggregated key* $\tilde{\mathsf{pk}}$*, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0, 1\}$.

That is, algorithm $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$ *can be written as* $\mathsf{VerAgg}(\mathsf{Agg}(\mathcal{P}), \mathsf{m}, \sigma)$.

4.4 Chopsticks: Fork-Free Two-Round Multi-Signatures

In this section, we present our constructions of two-round multi-signatures from [PW23a]. The section is structured in the following way: we first introduce two abstract building blocks, namely, linear function families and a special kind of commitment scheme. Then, given a linear function family LF and a commitment scheme CMT, we abstractly present and analyze two constructions of two-round multi-signatures ChopsKA[LF, CMT] (with key aggregation) and Chops[LF, CMT] (with tight security). We then show how to instantiate the building blocks, leading to our schemes Chopsticks I and Chopsticks II, respectively.

4.4.1 Building Blocks: Linear Functions and Special Commitments

Before we describe our constructions of two-round multi-signatures, we introduce two abstract building blocks that we will use and later instantiate. Our first building block, known as linear function families, has been used in previous works as an abstraction [HKL19, KLR21, CAHL+22a]. The properties that we define for linear function families are new and formalize under which condition a linear function family induces a lossy identification scheme [AFLT12]. Our second building block is a new kind of dual-mode commitment scheme with a weak equivocation feature.

Linear Function Families. To present our constructions in a modular way, we make use of the abstraction of linear function families. Our definition is close to previous definitions [HKL19, KLR21, CAHL+22a]. As it is not needed for our instantiations, we restrict our setting to vector spaces instead of pseudo modules.

Definition 4.4 (Linear Function Family)**.** *A linear function family (LFF) is a tuple of PPT algorithms* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs parameters* par*. We assume that* par *implicitly defines the following sets:*
  - *A set of scalars* $\mathcal{S}_{\mathsf{par}}$*, which forms a field.*
  - *A domain* $\mathcal{D}_{\mathsf{par}}$*, which forms a vector space over* $\mathcal{S}_{\mathsf{par}}$*.*
  - *A range* $\mathcal{R}_{\mathsf{par}}$*, which forms a vector space over* $\mathcal{S}_{\mathsf{par}}$*.*

  We omit the subscript par *if it is clear from the context, and naturally denote the operations of these fields and vector spaces by* $+$ *and* $\cdot$*. We assume that these operations can be evaluated efficiently.*
- $\mathsf{F}(\mathsf{par}, x) \to X$ *is deterministic, takes as input parameters* par*, an element* $x \in \mathcal{D}$*, and outputs an element* $X \in \mathcal{R}$*. For all parameters* par*,* $\mathsf{F}(\mathsf{par}, \cdot)$ *realizes a homomorphism, i.e.*

$$\forall s \in \mathcal{S}, x, y \in \mathcal{D} : \ \mathsf{F}(\mathsf{par}, s \cdot x + y) = s \cdot \mathsf{F}(\mathsf{par}, x) + \mathsf{F}(\mathsf{par}, y).$$

  We omit the input par *if it is clear from the context.*

We introduce the following definitions to formalize under which conditions a linear function family can be used to construct lossy identification [AFLT12]. Our constructions will rely on such linear function families. In a nutshell, we require that elements in the image of the function are indistinguishable from random elements in the range. Further, when turning the linear function into an identification scheme in a natural way, making the verifier accept is statistically infeasible if the public key is random. We also give a similar definition that captures this in the context of key aggregation.

Definition 4.5 (Key Indistinguishability)**.** *Let* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *be a linear function family. We say that* LF *satisfies key indistinguishability, if for any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}}(\lambda) := |\Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle|\ \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D},\ X := \mathsf{F}(x)\ \right] \\ - \Pr\left[\mathcal{A}(\mathsf{par}, X) = 1 \middle|\ \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}\ \right]|.\end{aligned}$$

Definition 4.6 (Lossy Soundness)**.** *Let* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *be a linear function family. We say that* LF *satisfies* ε_{l}*-lossy soundness, if for any unbounded algorithm* $\mathcal{A}$*, the following probability is at most* ε_{l}*:*

$$\Pr\left[\mathsf{F}(s) - c \cdot X = R \;\middle|\; \begin{array}{l}\mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}, \\ (St, R) \leftarrow \mathcal{A}(\mathsf{par}, X), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St, c)\end{array}\right].$$

Definition 4.7 (Aggregation Lossy Soundness)**.** *Let* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *be a linear function family. We say that* LF *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness, if for any unbounded algorithm* $\mathcal{A}$*, the following probability is at most* $\varepsilon_{\mathsf{al}}$*:*

$$\Pr\left[\mathsf{F}(s) - c \cdot \tilde{X} = R \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R}, \\ (St, (X_2, a_2), \ldots, (X_N, a_N)) \leftarrow \mathcal{A}(\mathsf{par}, X_1), \\ a_1 \xleftarrow{\$} \mathcal{S},\ (St', R) \leftarrow \mathcal{A}(St, a_1), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St', c),\ \tilde{X} := \sum_{i=1}^{N} a_i X_i \end{array}\right].$$

Weakly Equivocable Commitments. As our second building block, we define a special kind of commitment scheme. We will make use of such a scheme in our constructions of multi-signatures. Before we give the definition, we explain the desired properties at a high level. First of all, we want to be able to commit to elements $R \in \mathcal{R}$ in the range of a given linear function family. Second, we need the commitment scheme to be homomorphic in both messages and randomness, allowing us to aggregate commitments during the signing protocol. Third, we need a certain dual mode property, ensuring that we can set up keys either in a perfectly hiding or in a perfectly binding mode. This will allow us to make the commitment key for the forgery binding, while associating a equivocation trapdoor to the keys used to answer signing queries. Most importantly, we do not need a full-fledged equivocation feature. This is because we already know the structure of messages to which we want to open the commitment. Looking ahead, this is the reason we can instantiate the commitment in the DDH setting.

Game Q**-KEYDIST**$^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda)$	**Game** Q**-KEYDIST**$^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda)$
01 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}$	06 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}$
02 **if** $(\mathsf{par}, x) \notin \mathsf{Good}$: **return** 0	07 **if** $(\mathsf{par}, x) \notin \mathsf{Good}$: **return** 0
03 **for** $i \in [Q]$: $\mathsf{ck}_i \leftarrow \mathsf{BGen}(\mathsf{par})$	08 **for** $i \in [Q]$: $\mathsf{ck}_i \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}$
04 $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$	09 $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$
05 **return** β	10 **return** β

Figure 4.4: The games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ for a weakly equivocable commitment scheme CMT and an adversary $\mathcal{A}$.

Definition 4.8 (Weakly Equivocable Commitment Scheme)**.** *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family and* $\mathcal{G} = \{\mathcal{G}_{\mathsf{par}}\}, \mathcal{H} = \{\mathcal{H}_{\mathsf{par}}\}$ *be families of subsets of abelian groups with efficiently computable group operations* $\oplus$ *and* $\otimes$*, respectively. Let* $\mathcal{K} = \{\mathcal{K}_{\mathsf{par}}\}$ *be a family of sets. An* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* LF *with key space* $\mathcal{K}$*, randomness space* $\mathcal{G}$ *and commitment space* $\mathcal{H}$ *is a tuple of PPT algorithms* $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ *with the following syntax:*

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$ *takes as input parameters* par*, and outputs a key* $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$*.*
- $\mathsf{TGen}(\mathsf{par}, X) \to (\mathsf{ck}, \mathsf{td})$ *takes as input parameters* par*, and an element* $X \in \mathcal{R}$*, and outputs a key* $\mathsf{ck} \in \mathcal{K}_{\mathsf{par}}$ *and a trapdoor* td*.*
- $\mathsf{Com}(\mathsf{ck}, R; \varphi) \to \mathsf{com}$ *takes as input a key* ck*, an element* $R \in \mathcal{R}$*, and a randomness* $\varphi \in \mathcal{G}_{\mathsf{par}}$*, and outputs a commitment* $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$*.*
- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$ *takes as input a key* ck *and a trapdoor* td*, and outputs a commitment* $\mathsf{com} \in \mathcal{H}_{\mathsf{par}}$ *and a state* St*.*
- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$ *takes as input a state* St*, and an element* $c \in \mathcal{S}$*, and outputs randomness* $\varphi \in \mathcal{G}_{\mathsf{par}}$*, and elements* $R \in \mathcal{R}, s \in \mathcal{D}$*.*

We omit the subscript par *if it is clear from the context. Further, the algorithms are required to satisfy the following properties:*

- ***Homomorphism.*** *For all* $\mathsf{par} \in \mathsf{LF.Gen}(1^\lambda), \mathsf{ck} \in \mathcal{K}_{\mathsf{par}}, R_0, R_1 \in \mathcal{R}$ *and* $\varphi_0, \varphi_1 \in \mathcal{G}$*, the following holds:*

$$\mathsf{Com}(\mathsf{ck}, R_0; \varphi_0) \otimes \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1) = \mathsf{Com}(\mathsf{ck}, R_0 + R_1; \varphi_0 \oplus \varphi_1).$$

- ***Good Parameters.*** *There is a set* Good*, such that membership to* Good *can be decided in polynomial time, and*

$$\Pr\left[(\mathsf{par}, x) \notin \mathsf{Good} \mid \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x \xleftarrow{\$} \mathcal{D}\right] \leq \varepsilon_{\mathsf{g}},$$

- ***Uniform Keys.*** *For all* $(\mathsf{par}, x) \in \mathsf{Good}$*, the following distributions are identical:*

$$\{(\mathsf{par}, x, \mathsf{ck}) \mid \mathsf{ck} \xleftarrow{\$} \mathcal{K}_{\mathsf{par}}\} \text{ and } \{(\mathsf{par}, x, \mathsf{ck}) \mid (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x))\}.$$

- ***Weak Trapdoor Property.*** *For all* $(\mathsf{par}, x) \in \mathsf{Good}$*, and all* $c \in \mathcal{S}$*, the following distributions* $\mathcal{T}_0$ *and* $\mathcal{T}_1$ *have statistical distance at most* ε_{t}*:*

$$\mathcal{T}_0 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ (\mathsf{com}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}), \\ \mathsf{tr} \leftarrow \mathsf{TCol}(St, c) \end{array} \right\}$$

$$\mathcal{T}_1 := \left\{ (\mathsf{par}, \mathsf{ck}, \mathsf{td}, x, c, \mathsf{com}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, \mathsf{F}(x)) \\ r \xleftarrow{\$} \mathcal{D},\ R := \mathsf{F}(r),\ \varphi \xleftarrow{\$} \mathcal{G}, \\ \mathsf{com} := \mathsf{Com}(\mathsf{ck}, R; \varphi), \\ s := c \cdot x + r,\ \mathsf{tr} := (\varphi, R, s) \end{array} \right\}$$

- ***Multi-Key Indistinguishability.*** *For every* $Q = \mathsf{poly}(\lambda)$ *and any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}}(\lambda) := \left| \Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] - \Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda) \Rightarrow 1\right] \right|,$$

  where games $\mathbf{KEYDIST}_0, \mathbf{KEYDIST}_1$ *are defined in Figure 4.4.*

- ***Statistically Binding.*** *There exists some (unbounded) algorithm* Ext*, such that for every (unbounded) algorithm* $\mathcal{A}$ *the following probability is at most* ε_{b}*:*

$$\Pr\left[\mathsf{Com}(\mathsf{ck}, R'; \varphi') = \mathsf{com} \wedge R \neq R' \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par}),\ (\mathsf{com}, St) \leftarrow \mathcal{A}(\mathsf{ck}), \\ R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com}),\ (R', \varphi') \leftarrow \mathcal{A}(St) \end{array} \right].$$

4.4.2 Construction with Key Aggregation

In this section, we construct a two-round multi-signature scheme with key aggregation. Although the scheme will not be tight, the security proof will not use rewinding, leading to an acceptable security loss. For our scheme, we need a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$. It should satisfy key indistinguishability and aggregation lossy soundness. Further, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be an $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$-weakly equivocable commitment scheme for LF with key space $\mathcal{K}$ randomness space $\mathcal{G}$ and commitment space $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of our scheme $\mathsf{ChopsKA}[\mathsf{LF}, \mathsf{CMT}]$. Additionally, the scheme is presented as pseudocode in Figure A.1.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ defining the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a signer samples $\mathsf{sk} := x \xleftarrow{\$} \mathcal{D}$. The public key is $\mathsf{pk} := X := \mathsf{F}(x)$.

Key Aggregation. For N signers with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, the aggregated public key $\tilde{\mathsf{pk}}$ is computed (by algorithm Agg) as

$$\tilde{\mathsf{pk}} := \tilde{X} := \sum_{i=1}^{N} a_i \cdot X_i,$$

where $\mathsf{pk}_i = X_i$ and $a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$ for each $i \in [N]$.

Signing Protocol. Suppose N signers with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ want to sign a message $\mathsf{m} \in \{0,1\}^*$. We describe the signing protocol (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first signer, which holds a secret key $\mathsf{sk}_1 = x_1$ for public key $\mathsf{pk}_1 = X_1$.

1. *Commitment Phase.* The signer derives the aggregated public key $\tilde{\mathsf{pk}}$ as described above. Then, it derives a commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ depending on the message. The signer samples an element $r_1 \xleftarrow{\$} \mathcal{D}$ and sets $R_1 := \mathsf{F}(r_1)$. Next, it commits to R_1 by sampling $\varphi_1 \xleftarrow{\$} \mathcal{G}$ and setting $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$. Finally, it sends $\mathsf{pm}_{1,1} := \mathsf{com}_1$ to all signers.
2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output in the commitment phase. Here, message $\mathsf{pm}_{1,i}$ is sent by signer i and has the form $\mathsf{pm}_{1,i} = \mathsf{com}_i$. With this notation, the signer aggregates the commitments via $\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$. It computes the challenge c and coefficient a_1 via $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ and $a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$. Then, it computes the response s_1 as $s_1 := c \cdot a_1 \cdot x_1 + r_1$.

   Finally, the signer sends $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$ to all signers.
3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output in the response phase. Here, message $\mathsf{pm}_{2,i}$ is sent by signer i and has the form $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$. To compute the final signature, signers aggregate the responses and commitment randomness as follows:

$$s := \sum_{i \in [N]} s_i, \; \varphi := \bigoplus_{i \in [N]} \varphi_i.$$

   They output the final signature $\sigma := (\mathsf{com}, s, \varphi)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\mathsf{com}, s, \varphi)$ be a signature. To verify σ, we determine the aggregated public key $\tilde{\mathsf{pk}} = \tilde{X}$ as above. We reconstruct the commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$, and the challenge $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$. Then, we output 1 if and only if the following equation holds:

$$\mathsf{com} = \mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - c \cdot \tilde{X}; \varphi\right).$$

Completeness easily follows from the homomorphic properties of CMT and F. For a similar calculation, we refer to the proof of Lemma 4.2.

Lemma 4.1. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* LF*. Then* $\mathsf{ChopsKA}[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

Theorem 4.1. *Let* LF *be a linear function family that satisfies key indistinguishability and* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness. Let* CMT *be a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$*, and* $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{ChopsKA}[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}}, Q_{\mathsf{H}_a}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{ChopsKA}[\mathsf{LF},\mathsf{CMT}]}(\lambda) \leq{}& \varepsilon_{\mathsf{g}} + 4Q_S^2\varepsilon_{\mathsf{t}} + 4Q_S\varepsilon_{\mathsf{g}} + 4Q_S Q_{\mathsf{H}} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} \\ &+ \frac{4Q_S}{|\mathcal{R}|} + \frac{4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|} + 4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{al}} \\ &+ 4Q_S\left(\mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda)\right).\end{aligned}$$

Proof. Set $\mathsf{MS} := \mathsf{ChopsKA}[\mathsf{LF}, \mathsf{CMT}]$ and let $\mathcal{A}$ be a PPT algorithm. In the following, we present a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$ proving the statement. The games are presented in Figures A.2 and A.3.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is defined as $\mathbf{G}_0 := \mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$. To fix notation, we recall this game. First, the game samples $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ and a pair $(\mathsf{pk}, \mathsf{sk})$ with $\mathsf{sk} := x_1 \xleftarrow{\$} \mathcal{D}$ and $\mathsf{pk} := X_1 := \mathsf{F}(x)$. Then, $\mathcal{A}$ gets $\mathsf{par}, \mathsf{pk}$ as input, and access to oracles $\textsc{Sig}_0, \textsc{Sig}_1$. We omit signing oracle $\textsc{Sig}_2$. As in the proof of Theorem 4.2 this does not change the advantage of $\mathcal{A}$, as algorithm Sig_2 does not make any use of the secret key or a secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Further, $\mathcal{A}$ gets access to random oracles $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c$, simulated by the game in a lazy manner, using maps h, h_a, h_c, respectively. Finally, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game outputs 1 if and only if $\mathsf{pk}^* \in \mathcal{P}^*$,$(\mathcal{P}^*, \mathsf{m}^*) \notin \mathsf{Queried}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. We assume that the public key pk^* is equal to pk_1 for $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. We write $\sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$, and denote the aggregated key for $\mathcal{P}^*$ by $\tilde{\mathsf{pk}} := \tilde{X} := \mathsf{Agg}(\mathcal{P}^*)$. By definition, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Chops}[\mathsf{LF},\mathsf{CMT}]}(\lambda).$$

Game $\mathbf{G}_1$: In this game $\mathbf{G}_1$, we add a bad event and let the game abort if it occurs. Concretely, consider par, x_1 sampled by the game as described above, and let Good be as in the definition of a weakly equivocable commitment scheme. The game aborts if $(\mathsf{par}, x_1) \notin \mathsf{Good}$. By definition of the weakly equivocable commitment scheme, we have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \Pr[(\mathsf{par}, x_1) \notin \mathsf{Good}] \leq \varepsilon_{\mathrm{g}}.$$

Game $\mathbf{G}_2$: In this game $\mathbf{G}_2$, we introduce a map b, that maps inputs to random oracle H to bits. For each new input $(\tilde{\mathsf{pk}}, \mathsf{m})$ to H, the bit $b[\tilde{\mathsf{pk}}, \mathsf{m}]$ is sampled from a Bernoulli distribution with parameter $\gamma := 1/(Q_S + 1)$. Further, the game aborts if any of the follow occurs:

- For a signing query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ and $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$, it holds that $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, or
- for the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ and $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}^*)$, it holds that $b[\tilde{\mathsf{pk}}, \mathsf{m}^*] = 0$.

Note that the view of $\mathcal{A}$ is independent of the map b until an abort occurs. If the game does not abort, it is exactly like $\mathbf{G}_1$. Therefore, we can use the fact $(1 - 1/z)^z \geq 1/4$ for all $z \geq 2$ and get

$$\begin{aligned}
\Pr[\mathbf{G}_2 \Rightarrow 1] = \gamma(1-\gamma)^{Q_S} \cdot \Pr[\mathbf{G}_1 \Rightarrow 1] &= \frac{1}{Q_S+1}\left(1 - \frac{1}{Q_S+1}\right)^{Q_S} \cdot \Pr[\mathbf{G}_1 \Rightarrow 1] \\
&= \frac{1}{Q_S}\left(1 - \frac{1}{Q_S+1}\right)^{Q_S+1} \cdot \Pr[\mathbf{G}_1 \Rightarrow 1] \\
&\geq \frac{1}{4Q_S} \cdot \Pr[\mathbf{G}_1 \Rightarrow 1].
\end{aligned}$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, we change how random oracle H is executed. Consider a query $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ for which the hash value is not yet defined. Recall that in this case, a bit $b[\tilde{\mathsf{pk}}, \mathsf{m}]$ is sampled. Then, a commitment key ck has to be returned. In previous games, ck was sampled uniformly via $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$. Now, depending on this bit, we change how ck is computed. Namely, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, we sample $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_1)$ and store the trapdoor td in another map $tr[\tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{td}$. On the other hand, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we sample $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$.

We argue that games $\mathbf{G}_2$ and $\mathbf{G}_3$ are indistinguishable as follows. First, note that for case $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, the distribution of ck stays the same, because we can assume $(\mathsf{par}, x_1) \in \mathsf{Good}$ due to previous changes.

For the case $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we use a reduction $\mathcal{B}$ against the multi-key indistinguishability of CMT interpolating between $\mathbf{G}_2$ and $\mathbf{G}_3$. Precisely, $\mathcal{B}$ gets as input par, x_1 and Q_{H} commitment keys

$\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_\mathsf{H}}$. It simulates $\mathbf{G}_2$ for $\mathcal{A}$ with par while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ with $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$. In the end, it outputs whatever the game outputs. Clearly, we have

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \mathsf{Adv}^{Q_\mathsf{H}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_4$: Game $\mathbf{G}_4$ is as $\mathbf{G}_3$, but we change the execution of oracles $\textsc{Sig}_0, \textsc{Sig}_1$. Concretely, after this change, the secret key x_1 is no longer needed. Consider a query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$. Recall that for previous games, in such a query, a commitment key $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ is computed. Then, values r_1, φ_1 are sampled, and $R_1 := \mathsf{F}(r_1)$ and a commitment $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$ is computed. Later, in $\textsc{Sig}_1$, s_1 is computed as $s_1 := c \cdot a_1 \cdot x_1 + r_1$, where c and a_1 are output by H_c and H_a as in the scheme. Assuming that the game does not abort in this query, we can assume that $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, due to the change in $\mathbf{G}_2$. This means that the entry $\mathsf{td} := tr[\tilde{\mathsf{pk}}, \mathsf{m}]$ is defined, and was sampled together with ck using $\mathsf{TGen}(\mathsf{par}, X_1)$. We use this in game $\mathbf{G}_4$ as follows: The game no longer samples r_1 and φ_1. Instead, the commitment com_1 is computed via $(\mathsf{com}_1, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td})$. Later, in $\textsc{Sig}_1$, s_1 and φ_1 are computed using $(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St, c \cdot a_1)$. Applying the weak trapdoor property of CMT Q_S many times we obtain

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq Q_S \varepsilon_\mathsf{t}.$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we revert the change we introduced in $\mathbf{G}_1$. Concretely, the game no longer aborts if $(\mathsf{par}, x_1) \notin \mathsf{Good}$. As before, we get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \Pr[(\mathsf{par}, x_1) \notin \mathsf{Good}] \leq \varepsilon_\mathbf{g}.$$

Game $\mathbf{G}_6$: In game $\mathbf{G}_6$, we change how the public key X_1 is generated. Recall that it was generated as $\mathsf{F}(x_1)$ before, where $x_1 \xleftarrow{\$} \mathcal{D}$. In this game, we sample $X_1 \xleftarrow{\$} \mathcal{R}$ instead. Note that due to the change in $\mathbf{G}_4$, we do not need x_1 anymore. We sketch a simple reduction $\mathcal{B}'$ against the key indistinguishability of the linear function family LF to show indistinguishability of $\mathbf{G}_5$ and $\mathbf{G}_6$. Namely, $\mathcal{B}'$ gets par and X_1 as input, and simulates $\mathbf{G}_5$ for $\mathcal{A}$. In the end, it outputs whatever the game outputs. We have

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_7$: In game $\mathbf{G}_7$, we want to use the binding property of CMT. To do that, we introduce two changes. First, in oracle queries of the form $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ we first set $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$. Then, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, we simulate H_c as before. If $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we run the (unbounded) extraction algorithm Ext that exists according to the statistical binding property of CMT. Concretely, we run $R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ and store $r[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := R$, where r is another map. Then, we continue the simulation of H_c as before. Second, we change the winning condition of the game. Concretely, after $\mathcal{A}$ outputs forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, we parse $\sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$ and compute the aggregated key $\tilde{\mathsf{pk}} := \tilde{X} := \mathsf{Agg}(\mathcal{P}^*)$ as before. In addition to the verification steps that we had before, we now also compute $c^* := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*)$ and $R^* := \mathsf{F}(s^*) - c^* \cdot \tilde{X}$, and check if $R^* = r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$. If this does not hold, the game outputs 0.

Intuitively, these changes accomplish the following. The game extracts the values R from every commitment that is given by $\mathcal{A}$ via random oracle H_c, for which the commitment key ck was generated using algorithm BGen (cf. game $\mathbf{G}_3$). Then, we force the adversary into using the extracted value for its forgery.

Formally, we argue indistinguishability of $\mathbf{G}_6$ and $\mathbf{G}_7$ using an unbounded reduction to the statistical binding property of CMT. This reduction gets as input par and ck^*. It guesses $i_\mathsf{H} \xleftarrow{\$} [Q_\mathsf{H}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. The reduction simulates game $\mathbf{G}_7$ for $\mathcal{A}$ honestly, except for query i_H to random oracle H and query i_{H_c} to random oracle H_c. If it had to sample a $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ in the former query, it instead responds with ck^*. If it had to run Ext in the latter query, it outputs com to the experiment. If query i_H was used to derive the commitment key used in the forgery and query i_{H_c} was used to derive the challenge c^* for the forgery, and $R^* \neq r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$, then the reduction outputs $R^*; \varphi^*$. Otherwise, it outputs $\perp$. Clearly, if the reduction guesses the correct queries and the bad event separating $\mathbf{G}_6$ and

$\mathbf{G}_7$ occurs, then it breaks the statistical binding property. The view of $\mathcal{A}$ is as in $\mathbf{G}_7$, and independent of $(i_{\mathsf{H}}, i_{\mathsf{H}_c})$. Therefore, we obtain

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

Game $\mathbf{G}_8$: In game $\mathbf{G}_8$, we introduce another abort. Namely, the game aborts in a query $\mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk})$, for which $\mathsf{pk} = \mathsf{pk}_1$ and the hash value is not yet defined, but for $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$, there is some com, m such that $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ is already defined. The probability of this bad event is easily bounded. First, assume that $\mathsf{pk}_1 = X_1$ is not the zero vector in $\mathcal{R}$. The probability that X_1 is the zero vector is at most $1/|\mathcal{R}|$. Then, fix such a query $\mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk} = X_1)$, and any previous query to oracle H_c. The bad event can only occur if the input of the latter query starts with $\tilde{X}$, where $a_1 X_1 = \sum_{i=2}^{N} a_i X_i - \tilde{X}$. As X_1 is not the zero vector, the value $a_1 X_1$ is uniform over the span of X_1. Further the values on the right-hand side are fixed before a_1 is sampled, assuming that the bad event occurs. Thus, the probability of the bad event for this pair of queries is at most $1/|\mathcal{S}|$. We get

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq \frac{1}{|\mathcal{R}|} + \frac{Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|}.$$

Note that this change ensured that for the forgery output by $\mathcal{A}$, the query defining coefficient a_1 occurred before the query defining the challenge c^*.

To bound the probability that $\mathbf{G}_8$ outputs 1, we give an unbounded reduction from the aggregation lossy soundness of LF.

- The reduction gets as input parameters par and an element X_1. It samples $\hat{\imath}_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$ and $\hat{\imath}_{\mathsf{H}_a} \xleftarrow{\$} [Q_{\mathsf{H}_a}]$. Then, it simulates $\mathbf{G}_8$ honestly until $\mathcal{A}$ outputs a forgery, except for queries $\hat{\imath}_{\mathsf{H}_c}$ to oracle H_c and $\hat{\imath}_{\mathsf{H}_a}$ to oracle H_a.
- If the query $\hat{\imath}_{\mathsf{H}_c}$ to oracle H_c occurs before the query $\hat{\imath}_{\mathsf{H}_a}$ to oracle H_a, the reduction aborts its execution.
- Consider the query $\hat{\imath}_{\mathsf{H}_a}$ to oracle H_a. If the hash value is already defined, the reduction aborts its execution. Else, let this query be $\mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk})$. If $\mathsf{pk} \neq \mathsf{pk}_1$, the reduction aborts. Otherwise, it first parses $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ and queries $a_i := \mathsf{H}_a(\langle\mathcal{P}\rangle, \mathsf{pk}_i)$ for all $2 \leq i \leq N$. Then it outputs the pairs $(\mathsf{pk}_2, a_2), \ldots, (\mathsf{pk}_N, a_N)$ to the aggregation lossy soundness experiment. It gets as input a_1, sets $h_a[\langle\mathcal{P}\rangle, \mathsf{pk}] := a_1$, and continues the simulation as in $\mathbf{G}_8$.
- Consider the query $\hat{\imath}_{\mathsf{H}_c}$ to oracle H_c. Let this query be $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$. The reduction aborts its execution, if the hash value for this query is already defined. Else, it queries $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$. If $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, it aborts its execution. Otherwise, it runs $R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ as in $\mathbf{G}_8$. It outputs R to the aggregation lossy soundness experiment and obtains a value c in return. Then, it sets $h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := c$ and continues the simulation as in $\mathbf{G}_8$.
- When $\mathcal{A}$ outputs the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, the reduction runs all the verification steps in $\mathbf{G}_8$. Additionally, it checks if the value $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*)$ was defined during query $\hat{\imath}_{\mathsf{H}_c}$ to H_c, and the value $\mathsf{H}_a(\langle\mathcal{P}^*\rangle, \mathsf{pk}_1)$ was defined during $\hat{\imath}_{\mathsf{H}_a}$ to oracle H_a. If this is not the case, it aborts its execution. Otherwise, it returns $s := s^*$ to the aggregation lossy soundness experiment.

Clearly, unless the reduction aborts due to wrong guessing of the indices $\hat{\imath}_{\mathsf{H}_a}, \hat{\imath}_{\mathsf{H}_c}$, the view of $\mathcal{A}$ is exactly as in $\mathbf{G}_8$. Before any such abort, $\mathcal{A}$'s view is independent of the indices $\hat{\imath}_{\mathsf{H}_a}, \hat{\imath}_{\mathsf{H}_c}$. Also, it is clear that if the reduction does not abort, it outputs a valid solution to the aggregation lossy soundness experiment. Therefore, we get

$$\Pr[\mathbf{G}_8 \Rightarrow 1] \leq Q_{\mathsf{H}_a} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{al}},$$

and the statement is proven. □

4.4.3 Tight Construction

In this section, we present a tightly secure two-round multi-signature scheme $\mathsf{Chops}[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. Let us first describe the building blocks that we need. We make use of a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$, for which we assume it satisfies key indistinguishability and lossy soundness. Also, let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be an $(\varepsilon_{\mathtt{b}}, \varepsilon_{\mathtt{g}}, \varepsilon_{\mathtt{t}})$-weakly equivocable commitment scheme for LF with key space $\mathcal{K}$ randomness space $\mathcal{G}$ and commitment space $\mathcal{H}$. We make use of random oracles $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$. We give a verbal description of the scheme. In addition, we present the scheme as pseudocode in Figure A.4.

Setup and Key Generation. The public parameters of the scheme are $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. They define the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. To generate a key (algorithm Gen), a signer samples $x_0, x_1 \xleftarrow{\$} \mathcal{D}$ and a seed $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. Then, it sets

$$\mathsf{sk} := (x_0, x_1, \mathsf{seed}), \quad \mathsf{pk} := (X_0, X_1) := (\mathsf{F}(x_0), \mathsf{F}(x_1)).$$

Signing Protocol. Suppose N signers with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ want to sign a message $\mathsf{m} \in \{0,1\}^*$. We describe the signing protocol (algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$) from the perspective of the first signer, which holds a secret key $\mathsf{sk}_1 = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ for public key $\mathsf{pk}_1 = (X_{1,0}, X_{1,1})$.

1. *Commitment Phase.* The signer derives commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$ and $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$ depending on the message. Then, it computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. It samples two elements $r_{1,0}, r_{1,1} \xleftarrow{\$} \mathcal{D}$ and sets

   $$R_{1,0} := \mathsf{F}(r_{1,0}), \quad R_{1,1} := \mathsf{F}(r_{1,1}).$$

   Next, it commits to the resulting elements by sampling $\varphi_{1,0}, \varphi_{1,1} \xleftarrow{\$} \mathcal{G}$ and setting

   $$\mathsf{com}_{1,0} := \mathsf{Com}(\mathsf{ck}_0, R_{1,0}; \varphi_{1,0}), \quad \mathsf{com}_{1,1} := \mathsf{Com}(\mathsf{ck}_1, R_{1,1}; \varphi_{1,1}).$$

   Finally, it sends $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1})$ to all signers.

2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output in the commitment phase. Here, message $\mathsf{pm}_{1,i}$ is sent by signer i and has the form $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1})$. With this notation, the signer sets $B := b_1 \ldots b_N \in \{0,1\}^N$. Then, it aggregates the commitments via

   $$\mathsf{com}_0 := \bigotimes_{i \in [N]} \mathsf{com}_{i,0}, \quad \mathsf{com}_1 := \bigotimes_{i \in [N]} \mathsf{com}_{i,1}.$$

   It computes signer specific challenges via

   $$c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1),$$

   and the responses as

   $$s_{1,0} := c_{1,0} \cdot x_{1,b_1} + r_{1,0}, \quad s_{1,1} := c_{1,1} \cdot x_{1,1-b_1} + r_{1,1}.$$

   Observe that the bit b_1 determines the link between the responses, challenges, and public keys. Finally, the signer sends $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ to all signers.

3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output in the response phase. Here, message $\mathsf{pm}_{2,i}$ is sent by signer i and has the form $\mathsf{pm}_{2,i} = (s_{i,0}, s_{i,1}, \varphi_{i,0}, \varphi_{i,1})$. To compute the final signature, signers aggregate the responses and commitment randomness as follows:

   $$s_0 := \sum_{i \in [N]} s_{i,0}, \; s_1 := \sum_{i \in [N]} s_{i,1}, \; \varphi_0 := \bigoplus_{i \in [N]} \varphi_{i,0}, \; \varphi_1 := \bigoplus_{i \in [N]} \varphi_{i,1}.$$

   They define $\sigma_0 := (\mathsf{com}_0, \varphi_0, s_0), \sigma_1 := (\mathsf{com}_1, \varphi_1, s_1)$ and output the final signature $\sigma := (\sigma_0, \sigma_1, B)$.

Verification. For verification (algorithm Ver), let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\sigma_0, \sigma_1, B)$ be a signature. To verify σ, we write $B = b_1 \ldots b_N$, $\sigma_0 = (\mathsf{com}_0, \varphi_0, s_0)$ and $\sigma_1 = (\mathsf{com}_1, \varphi_1, s_1)$. Further, we write the public keys pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. We reconstruct the commitment keys $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$, $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$, and the signer specific challenges

$$c_{i,0} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0), \quad c_{i,1} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1).$$

Then, we output 1 if and only if the following two equations hold:

$$\mathsf{com}_0 = \mathsf{Com}\left(\mathsf{ck}_0, \mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i}; \varphi_0\right)$$
$$\mathsf{com}_1 = \mathsf{Com}\left(\mathsf{ck}_1, \mathsf{F}(s_1) - \sum_{i=1}^{N} c_{i,1} \cdot X_{i,1-b_i}; \varphi_1\right).$$

Lemma 4.2. *Let* LF *be a linear function family. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-special commitment scheme for* LF*. Then* Chops[LF, CMT] *is complete.*

Proof. Consider the variables given in verification and an honest execution of the protocol. Concretely, let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be a set of public keys, $\mathsf{m} \in \{0,1\}^*$ be a message, and $\sigma = (\sigma_0, \sigma_1, B)$ be a signature computed by an honest execution of the signing protocol specified by algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$. Write $B = b_1 \ldots b_N$, $\sigma_0 = (\mathsf{com}_0, \varphi_0, s_0)$ and $\sigma_1 = (\mathsf{com}_1, \varphi_1, s_1)$. Write the public keys pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. Then, we can use the homomorphic properties of F to obtain

$$\begin{aligned}
\mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i} &= \mathsf{F}\left(\sum_{i=1}^{N} s_{i,0}\right) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i} \\
&= \sum_{i=1}^{N} \mathsf{F}(s_{i,0}) - c_{i,0} \cdot \mathsf{F}(x_{i,b_i}) \\
&= \sum_{i=1}^{N} \mathsf{F}(s_{i,0} - c_{i,0} \cdot x_{i,b_i}) = \sum_{i=1}^{N} \mathsf{F}(r_{i,0}) = \sum_{i=1}^{N} R_{i,0}.
\end{aligned}$$

Using this, the homomorphic properties of Com, and the definition of φ_0, it follows that

$$\begin{aligned}
\mathsf{Com}\left(\mathsf{ck}_0, \mathsf{F}(s_0) - \sum_{i=1}^{N} c_{i,0} \cdot X_{i,b_i}; \varphi_0\right) &= \mathsf{Com}\left(\mathsf{ck}_0, \sum_{i=1}^{N} R_{i,0}; \bigoplus_{i=1}^{N} \varphi_{i,0}\right) \\
&= \bigotimes_{i=1}^{N} \mathsf{Com}\left(\mathsf{ck}_0, R_{i,0}; \varphi_{i,0}\right) \\
&= \bigotimes_{i=1}^{N} \mathsf{com}_{i,0} = \mathsf{com}_0.
\end{aligned}$$

This shows that the first verification equation holds. The proof for the second equation is similar. □

Theorem 4.2. *Let* LF *be a linear function family that satisfies key indistinguishability and* ε_l*-lossy soundness. Let* CMT *be a* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-weakly equivocable commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}, \mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* Chops[LF, CMT] *is* MS-EUF-CMA *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ that makes at most $Q_{\mathsf{H}}, Q_{\mathsf{H}_b}, Q_{\mathsf{H}_c}, Q_S$ queries to oracles $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c, \textsc{Sig}_0$, respectively, there are PPT algorithms $\mathcal{B}, \mathcal{B}'$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ and

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Chops}[\mathsf{LF},\mathsf{CMT}]}(\lambda) \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda} + 4\varepsilon_{\mathsf{g}} + 2Q_S\varepsilon_{\mathsf{t}} + 2Q_{\mathsf{H}}Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} + 2Q_{\mathsf{H}_c}\varepsilon_{\mathsf{I}} + 2 \cdot \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Proof. Set $\mathsf{MS} := \mathsf{Chops}[\mathsf{LF}, \mathsf{CMT}]$. Let $\mathcal{A}$ be a PPT algorithm as in the statement. We prove the claim via a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$. The games are formally presented in Figures A.5 to A.7, and we describe and analyze them verbally.

Game $\mathbf{G}_0$: We define $\mathbf{G}_0$ to be exactly as $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}$, with the following modification: The adversary $\mathcal{A}$ does not get access to oracle $\textsc{Sig}_2$. Note that in MS, algorithm Sig_2 does not make any use of the secret key or a secret state and can be publicly run using the messages output in Sig_0 and Sig_1. Therefore, for any adversary in the original game, there is an adversary in game $\mathbf{G}_0$ that simulates oracle $\textsc{Sig}_2$ and has the same advantage.

Before we proceed, let us describe game $\mathbf{G}_0$ in more detail to fix some notation. In the beginning, the game samples parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$. It also samples a public key $\mathsf{pk}^* = (X_{1,0}, X_{1,1})$ $= (\mathsf{F}(x_{1,0}), \mathsf{F}(x_{1,1}))$ for a secret key $\mathsf{sk}^* = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ with $x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D}, \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$. Then, it runs $\mathcal{A}$ on input $\mathsf{par}, \mathsf{pk}^*$ with access to the following oracles:

- Signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$: The oracles simulate algorithms Sig_0 and Sig_1 on secret key sk^*, respectively. Here, $\mathcal{A}$ can submit a query $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$ to start a new interaction in which message m is signed for public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. We assume that $\mathsf{pk}^* = \mathsf{pk}_1$, and the oracle adds $(\mathcal{P}, \mathsf{m})$ to a list $\mathsf{Queried}$.
- Random oracles $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c$: The random oracles H, H_c are simulated honestly via lazy sampling. To this end, the game holds maps h, h_c that map the inputs of the respective random oracles to their outputs. Random oracle H_b, however, is simulated by forwarding the query to an internal oracle $\bar{\mathsf{H}}_b$ with the same interface. This oracle holds a similar map $\bar{h}_b$, is kept internally by the game, and is not provided to the adversary. Looking ahead, this indirection allows us to distinguish queries to H_b that some of the following games issue from the queries that the adversary issues.

In the end, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game outputs 1 if and only if $\mathsf{pk}^* \in \mathcal{P}^*$, $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathsf{Queried}$, and $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. Without loss of generality, we assume that the public key pk^* is equal to pk_1 for $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. To fix notation, write $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \ldots b_N^*$ and $\sigma_0^* = (\mathsf{com}_0^*, \varphi_0^*, s_0^*), \sigma_1^* = (\mathsf{com}_1^*, \varphi_1^*, s_1^*)$. Clearly, we have

$$\Pr[\mathbf{G}_0 \Rightarrow 1] = \mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Chops}[\mathsf{LF},\mathsf{CMT}]}(\lambda).$$

Game $\mathbf{G}_1$: In game $\mathbf{G}_1$, we add an abort. Namely, the game sets $\mathsf{bad} := 1$, and aborts, if the adversary makes a random oracle query $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$. Note that this does not include the queries that are made by the game itself, as these are done using oracle $\bar{\mathsf{H}}_b$ directly. As the only information about seed_1 that $\mathcal{A}$ gets are the values of $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$, and seed_1 is sampled uniformly at random from $\{0,1\}^\lambda$, we can upper bound the probability of bad by $Q_{\mathsf{H}_b}/2^\lambda$. Therefore, we have

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \Pr[\mathsf{bad}] \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda}.$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, we restrict the winning condition. Namely, the game outputs 0, if the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ output by $\mathcal{A}$ satisfies $b_1^* \neq 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$. Recall that b_1^* is the bit related to $\mathsf{pk}_1 = \mathsf{pk}^*$ that is included in the signature σ^*. Assuming $\mathbf{G}_1$ outputs 1, we know that $(\mathcal{P}^*, \mathsf{m}^*) \notin$

Queried. Therefore, $\mathcal{A}$ can only get information about the bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}^*\rangle, \mathsf{m}^*)$, if it queries the wrapper random oracle H_b at this position. However, in this case $\mathbf{G}_1$ would set $\mathsf{bad} := 1$ and abort. Thus, the view of $\mathcal{A}$ is independent of bit $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}^*\rangle, \mathsf{m}^*)$. We obtain

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr\left[\mathbf{G}_1 \Rightarrow 1 \wedge b_1^* = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}^*\rangle, \mathsf{m}^*)\right] = \frac{1}{2}\Pr[\mathbf{G}_1 \Rightarrow 1].$$

Game $\mathbf{G}_3$: In game $\mathbf{G}_3$, the game aborts if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$, where Good is as in the definition of a weakly equivocable commitment scheme. It is clear that

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \Pr[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_4$: In game $\mathbf{G}_4$, we change the behavior of random oracle H. Recall that before, to answer a query $\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ for which the hash value has not been defined, a key $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$ was sampled and returned. In this game, the oracle instead distinguishes two cases. In the first case, if $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$, the game samples $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1})$. It also stores $tr[\langle\mathcal{P}\rangle, \mathsf{m}] := \mathsf{td}$, where tr is a map. In the second case, if $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$, it samples $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. In both cases, ck is returned as before. To see that $\mathbf{G}_3$ and $\mathbf{G}_4$ are indistinguishable, we first note that for the first case, the distribution of ck stays the same. This is because we can assume $(\mathsf{par}, x_{1,1}) \in \mathsf{Good}$ due to the previous change. The keys returned in the second case are indistinguishable by the multi-key indistinguishability of CMT. More precisely, we give a reduction $\mathcal{B}$ against the multi-key indistinguishability of CMT that interpolates between $\mathbf{G}_3$ and $\mathbf{G}_4$. The reduction gets as input $\mathsf{par}, x_{1,1}$ and Q_{H} commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$. It simulates $\mathbf{G}_3$ for $\mathcal{A}$ with par while embedding the commitment keys in random oracle responses for queries $\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ with $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. In the end, it outputs whatever the game outputs[7] . We have

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_5$: In game $\mathbf{G}_5$, we change the signing oracles $\textsc{Sig}_0, \textsc{Sig}_1$. Our goal is to eliminate the use of the secret key component $x_{1,1}$. Recall that in previous games, oracle $\textsc{Sig}_0$ derived a bit $b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$ and sampled random $r_{1,0}, r_{1,1}$ and $\varphi_{1,0}, \varphi_{1,1}$. Then, these were used to compute commitments $\mathsf{com}_{1,0}, \mathsf{com}_{1,1}$, which where then output together with b_1. Then, in oracle $\textsc{Sig}_1$ the values $s_{1,0}, s_{1,1}$ were computed using the secret keys $x_{1,b_1}, x_{1,1-b_1}$, respectively.

In this game, we change how the commitment $\varphi_{1,1-b_1}$ and the value $s_{1,1-b_1}$ is computed to eliminate the dependency on $x_{1,1}$. Namely, in oracle $\textsc{Sig}_0$, we do not compute $r_{1,1-b_1}, \varphi_{1,1-b_1}$ and $R_{1,1-b_1}$ anymore. Instead, we compute the commitment $\mathsf{com}_{1,1-b_1}$ via

$$\mathsf{td} := tr[\langle\mathcal{P}\rangle, , \mathsf{m}], \quad (\mathsf{com}_{1,1-b_1}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_{1-b_1}, \mathsf{td}).$$

Note that $\mathsf{ck}_{1-b_1} = \mathsf{H}(1 - b_1, \langle\mathcal{P}\rangle, \mathsf{m})$, and therefore ck_{1-b_1} and td were generated using algorithm $\mathsf{TGen}(\mathsf{par}, X_{1,1})$ due to the change in $\mathbf{G}_4$. Later, in oracle $\textsc{Sig}_1$, we derive

$$(\varphi_{1,1-b_1}, R_{1-b_1}, s_{1,1-b_1}) \leftarrow \mathsf{TCol}(St, c_{1,1-b_1}).$$

Then, message $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ is output as before.

We can easily argue indistinguishability by using the weak trapdoor property of CMT Q_{S_0} many times and get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq Q_S \varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_6$: Here we do not abort if $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$ anymore. That is, we revert the change introduced in $\mathbf{G}_3$. It is clear that

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \Pr[(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}] \leq \varepsilon_{\mathsf{g}}.$$

[7] Note that at this point, it was important that we introduced the oracle $\bar{\mathsf{H}}_b$. This is because otherwise, if we queried $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$ in oracle H, game $\mathbf{G}_3$ would always output 0 and the games would not be indistinguishable.

Game $\mathbf{G}_7$: In game $\mathbf{G}_7$, we change how the public key component $X_{1,1}$ is computed. Recall that before, $X_{1,1}$ is computed as $X_{1,1} := \mathsf{F}(x_{1,1})$ for $x_{1,1} \xleftarrow{\$} \mathcal{D}$. Also, note that due to the previous changes, the value $x_{1,1}$ is not used anymore. In $\mathbf{G}_7$, we sample $X_{1,1} \xleftarrow{\$} \mathcal{R}$. A direct reduction $\mathcal{B}'$ against the key indistinguishability of the linear function family LF shows indistinguishability of $\mathbf{G}_6$ and $\mathbf{G}_7$. Concretely, $\mathcal{B}'$ gets par and $X_{1,1}$ as input, and simulates $\mathbf{G}_6$ for $\mathcal{A}$. In the end, it outputs whatever the game outputs. We have

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_8$: In game $\mathbf{G}_8$, we change how queries to random oracle H_c are answered. Concretely, consider a query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$ with $\mathsf{pk} = \mathsf{pk}^*$ and $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. For these queries, the game now runs $R \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$ and stores $r[\mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] := R$, where r is another map. Here, Ext is the (unbounded) extractor for the statistical binding property of CMT. The rest of the oracle does not change. Note that for b of this form, the value $\mathsf{ck} = \mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ is sampled as $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ (cf. $\mathbf{G}_4$). We also slightly change the winning condition of the game. Namely, in $\mathbf{G}_8$, consider the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ with $\sigma^* = (\sigma_0^*, \sigma_1^*, B^*)$, $B^* = b_1^* \dots b_N^*$, and let $R_0^*, R_1^* \in \mathcal{R}$ be the values that are computed during the execution of $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. The game returns 0 if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$.

We claim that indistinguishability of $\mathbf{G}_7$ and $\mathbf{G}_8$ can be argued using the statistical binding property of CMT. To see this, assume that $\mathbf{G}_7$ outputs 1. Then, due to the change in $\mathbf{G}_2$, we know that $1 - b_1^* = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}^*\rangle, \mathsf{m}^*)$. Therefore, in the corresponding query $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*, 1 - b_1^*)$ algorithm Ext was run and the value $r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$ is defined. Next, by definition of Ver, we have $\mathsf{Com}(\mathsf{ck}_{1-b_1^*}, R^*_{1-b_1^*}; \varphi^*_{1-b_1^*}) = \mathsf{com}^*_{1-b_1^*}$. Therefore, if $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$, we have a contradiction to the statistical binding property of CMT. More precisely, we sketch an (unbounded) reduction from the statistical binding property. Namely, this reduction gets as input par and a commitment key ck^*. Then, the reduction guesses $i_{\mathsf{H}} \xleftarrow{\$} [Q_{\mathsf{H}}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. It simulates game $\mathbf{G}_8$ honestly, except for query i_{H} to random oracle H and query i_{H_c} to random oracle H_c. If it had to sample a $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$ in the former query, it instead responds with ck^*. Similarly, if it had to run Ext in the latter query, it outputs com to the binding experiment. If these queries are the queries of interest (i.e., query i_{H} was used to derive $\mathsf{ck}_{1-b_1^*}$ and query i_{H_c} was used to derive $c^*_{1,1-b_1^*}$) for the forgery, and $R^*_{1-b_1^*} \neq r[\mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle\mathcal{P}^*\rangle, B^*]$, then the reduction outputs $R^*_{1-b_1^*}; \varphi^*_{1-b_1^*}$. Otherwise, it outputs $\perp$. It is easy to see that if the reduction guesses the correct queries and the bad event separating $\mathbf{G}_7$ and $\mathbf{G}_8$ occurs, then it breaks the statistical binding property. As the view of $\mathcal{A}$ is as in $\mathbf{G}_8$, and independent of $(i_{\mathsf{H}}, i_{\mathsf{H}_c})$, we obtain

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

Finally, we use lossy soundness of LF to bound the probability that $\mathbf{G}_8$ outputs 1. To do that, we give an unbounded reduction from the lossy soundness experiment, which is as follows.

- The reduction gets par, $X_{1,1}$ as input. It samples $\hat{i} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. Then, it simulates $\mathbf{G}_8$ honestly until $\mathcal{A}$ outputs a forgery, except for query $\hat{i}$ to oracle H_c.
- Consider this query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$. The reduction aborts its execution, if the hash value for this query is already defined, or if $\mathsf{pk} \neq \mathsf{pk}^* \vee b \neq \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. Otherwise, it runs $\hat{R} \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$ as in $\mathbf{G}_8$. Then, it parses $\mathcal{P} = \{\mathsf{pk}_1, \dots, \mathsf{pk}_N\}$ and $B = b_1 \dots b_N$. It parses $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$ for each $i \in [N]$, and it sets $c_{i,b} = \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$ for each $i \in [N] \setminus \{1\}$. Next, it defines

$$R := \hat{R} + \sum_{i=2}^{N} c_{i,b} \cdot X_{i,\hat{b}_i},$$

where $\hat{b}_i := (b + b_i) \mod 2$. It outputs R to the lossy soundness game and obtains a value c in return. Then, it sets $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b] := c$ and continues the simulation as in $\mathbf{G}_8$.

- When the reduction gets the forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ from $\mathcal{A}$, it runs all the verification steps in $\mathbf{G}_8$. Additionally, it checks if the value $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1-b_1^*)$ was defined during query $\hat{i}$ to H_c. If this is not the case, it aborts its execution. Otherwise, it returns $s := s^*_{1-b_1^*}$ to the lossy soundness game.

It is clear that the view of $\mathcal{A}$ is independent of the index $\hat{i}$ until a potential abort of the reduction. Also, if the reduction does not abort its execution, it perfectly simulates game $\mathbf{G}_8$ for $\mathcal{A}$. Thus, it remains to show that if $\mathbf{G}_8$ outputs 1, then the values output by the reduction satisfy $\mathsf{F}(s) - c \cdot X_{1,1} = R$. Once we have shown this, it follows that

$$\Pr[\mathbf{G}_8 \Rightarrow 1] \leq Q_{\mathsf{H}_c} \varepsilon_{\mathsf{I}}.$$

To show the desired property, assume that the reduction does not abort and $\mathbf{G}_8$ outputs 1. Then, define $\hat{b}_i^* = (1 - b_1^* + b_i) \mod 2$ for all $i \in [N]$. Note that $\hat{b}_{\hat{i}}^* = 1$. Due to the change in $\mathbf{G}_2$, we have

$$b = 1 - b_1^* = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*).$$

As the reduction guessed the right query and does not abort, we have

$$c^*_{1,1-b_1^*} = \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*_{1-b_1^*}, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1-b_1^*) = c.$$

Due to the change in $\mathbf{G}_8$, we have

$$\mathsf{F}(s^*_{1-b_1^*}) - \sum_{i=1}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} = R^*_{1-b_1^*} = \hat{R}.$$

Therefore, we have

$$\begin{aligned}
\mathsf{F}(s) - c \cdot X_{1,1} &= \mathsf{F}(s^*_{1-b_1^*}) - c^*_{1,1-b_1^*} \cdot X_{1,1} \\
&= \mathsf{F}(s^*_{1-b_1^*}) - \sum_{i=1}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} + \sum_{i=2}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} \\
&= \hat{R} + \sum_{i=2}^{N} c^*_{i,1-b_1^*} \cdot X_{i,\hat{b}_i^*} = R.
\end{aligned}$$

Concluded. □

4.4.4 Instantiation

In this section, we show how to instantiate the building blocks that are needed for our constructions in the previous section. Concretely, we give a linear function family and a commitment scheme based on the DDH assumption.

Linear Function Family. We make use of the well-known [KW03, KMP16] linear function family $\mathsf{LF}_{\mathsf{DDH}} = (\mathsf{Gen}, \mathsf{F})$ based on the DDH assumption. Precisely, let GGen be an algorithm that on input 1^λ outputs the description of a prime order group $\mathbb{G}$ of order p with generator g. Then, Gen runs GGen and outputs[8] $\mathsf{par} := (g, h) \in \mathbb{G}^2$ for $h \xleftarrow{\$} \mathbb{G}$. Then, the set of scalars, domain, range, and function $\mathsf{F}(\mathsf{par}, \cdot)$ are given as follows:

$$\mathcal{S} := \mathbb{Z}_p, \quad \mathcal{D} := \mathbb{Z}_p, \quad \mathcal{R} := \mathbb{G} \times \mathbb{G}, \quad \mathsf{F}(\mathsf{par}, x) := (g^x, h^x).$$

It is easily verified that this constitutes a linear function family. We show that it satisfies key indistinguishability, lossy soundness, and aggregation lossy soundness.

[8] We omit the description of $\mathbb{G}$ from par to make the presentation concise.

Lemma 4.3. *Assuming that the* DDH *assumption holds relative to* GGen*, the linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies key indistinguishability. Concretely, for any PPT algorithm* $\mathcal{A}$ *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda).$$

Proof. The proof is trivial: key indistinguishability matches exactly the DDH assumption. □

Lemma 4.4. *The linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies* ε_{l}*-lossy soundness with* $\varepsilon_{\mathsf{l}} \leq 3/p$.

Proof. We have to bound the probability

$$\Pr\left[(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}) = (R_1, R_2) \;\middle|\; \begin{array}{l} (g,h) \leftarrow \mathsf{Gen}(1^\lambda),\ (X_1, X_2) \xleftarrow{\$} \mathbb{G}^2, \\ (St, (R_1, R_2)) \leftarrow \mathcal{A}((g,h),(X_1,X_2)), \\ c \xleftarrow{\$} \mathbb{Z}_p,\ s \leftarrow \mathcal{A}(St, c) \end{array}\right].$$

The probability that $h = g^0$ is at most $1/p$. Thus, we assume that h is a generator of $\mathbb{G}$. Write $X_1 = g^{x_1}$ and $X_2 = h^{x_2}$. With probability at most $1/p$ we have $x_1 = x_2$. Assume that $x_1 \neq x_2$. We claim that with these assumptions, the probability that we have to bound is at most $1/p$. To see this, assume that there is some (R_1, R_2) such that there exist two different $c \neq c'$ in $\mathbb{Z}_p$, such that there exists a $s, s' \in \mathbb{Z}_p$ with

$$\left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right) = (R_1, R_2) \text{ and } \left(g^{s'} \cdot X_1^{-c'}, h^{s'} \cdot X_2^{-c'}\right) = (R_1, R_2).$$

Then, we can combine both equations and rearrange terms to get

$$\left(g^{(s-s')/(c-c')}, h^{(s-s')/(c-c')}\right) = (X_1, X_2),$$

contradicting our assumption that $x_1 \neq x_2$. The claim follows. □

Lemma 4.5. *The linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness with* $\varepsilon_{\mathsf{al}} \leq 4/p$.

Proof. Let $\mathcal{A}$ be any unbounded algorithm. We have to bound the probability that

$$\left(g^s \cdot \tilde{X}_1^{-c}, h^s \cdot \tilde{X}_2^{-c}\right) = (R_1, R_2),$$

where we consider the following experiment. First, $(g,h) \leftarrow \mathsf{Gen}(1^\lambda)$, $(X_1, X_2) \xleftarrow{\$} \mathbb{G}^2$ is sampled and g, h, X_1, X_2 are given to $\mathcal{A}$. Then, $\mathcal{A}$ outputs pairs of group elements and exponents $((X_{2,1}, X_{2,2}), a_2)$, $\ldots, ((X_{N,1}, X_{N,2}), a_N)$. Next, exponent $a_1 \xleftarrow{\$} \mathbb{Z}_p$ are sampled and $\tilde{X}_1, \tilde{X}_2$ are defined as

$$\left(\tilde{X}_1, \tilde{X}_2\right) := \left(\prod_{i=1}^{N} X_{i,1}^{a_i}, \prod_{i=1}^{N} X_{i,2}^{a_i}\right).$$

Then, $\mathcal{A}$ outputs (R_1, R_2) on input a_1. A challenge $c \xleftarrow{\$} \mathbb{Z}_p$ is sampled and $\mathcal{A}$ outputs s on input c.

The probability that $h = g^0$ is at most $1/p$. Thus, we assume that h is a generator of $\mathbb{G}$. Looking at the proof of Lemma 4.4, we see that it is sufficient to argue that with high probability, $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not of the form $(g^{\tilde{x}}, h^{\tilde{x}})$ for any $\tilde{x} \in \mathbb{Z}_p$. In other words, we have to show that with high probability, the pair $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not in the image of F. Conditioned on that, as in the proof of Lemma 4.4, the probability above can be bounded by $1/p$.

To show this, we fix the exponents $x_{i,j} \in \mathbb{Z}_p$ such that $X_{i,1} = g^{x_{i,1}}$ and $X_{i,2} = h^{x_{i,2}}$. The probability that $x_{1,1} = x_{1,2}$ is at most $1/p$. From now on, we condition on $x_{1,1} \neq x_{1,2}$. The pair $\left(\tilde{X}_1, \tilde{X}_2\right)$ is not in the image of F if and only if

$$\sum_{i=1}^{N} a_i x_{i,1} = \sum_{i=1}^{N} a_i x_{i,2}.$$

This is equivalent to

$$a_1 = \frac{\sum_{i=2}^{N} a_i x_{i,2} - \sum_{i=2}^{N} a_i x_{i,2}}{x_{1,1} - x_{1,2}}.$$

As a_1 is sampled uniformly over $\mathbb{Z}_p$ after $\mathcal{A}$ choses the $x_{i,j}$ and a_i, $i > 2$, the above holds with probability at most $1/p$, and the claim follows. □

Commitment Scheme. We give a weakly equivocable trapdoor commitment scheme $\mathsf{CMT}_{\mathsf{DDH}} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ for the linear function family $\mathsf{LF}_{\mathsf{DDH}}$. For given parameters of $\mathsf{LF}_{\mathsf{DDH}}$, the commitment scheme has key space $\mathcal{K} := \mathbb{G}^{3\times 3}$ and message space $\mathcal{D} = \mathbb{G} \times \mathbb{G}$. It has randomness space $\mathcal{G} = \mathbb{Z}_p^3$ and commitment space $\mathcal{H} = \mathbb{G}^3$. Both are associated with the natural componentwise group operations. We describe the algorithms of the scheme verbally.

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$: Sample $g_1, g_2, g_3 \xleftarrow{\$} \mathbb{G}$, and $a, b \xleftarrow{\$} \mathbb{Z}_p$, and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} g_1 & g_1^a & g_1^b \\ g_2 & g_2^a & g_2^b \\ g_3 & g_3^a & g_3^b \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

- $\mathsf{TGen}(\mathsf{par}, X = (X_1, X_2)) \to (\mathsf{ck}, \mathsf{td})$: Sample $d_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for all $(i,j) \in [3] \times [3]$ and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} := \begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} \\ X_1^{d_{2,1}} & X_1^{d_{2,2}} & X_1^{d_{2,3}} \\ X_2^{d_{3,1}} & X_2^{d_{3,2}} & X_2^{d_{3,3}} \end{pmatrix} \in \mathbb{G}^{3\times 3}.$$

  Next, set

$$\mathsf{td} := (\mathbf{D}, X_1, X_2), \text{ for } \mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ d_{3,1} & d_{3,2} & d_{3,3} \end{pmatrix} \in \mathbb{Z}_p^{3\times 3}.$$

- $\mathsf{Com}(\mathsf{ck}, R = (R_1, R_2); \varphi) \to \mathsf{com}$: Let $\varphi = (\alpha, \beta, \gamma) \in \mathbb{Z}_p^3$. Compute

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_2 \cdot & A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix}.$$

- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$: Sample $\tau, \rho_1, \rho_2, s \xleftarrow{\$} \mathbb{Z}_p$. Set $St := (\mathsf{td}, \tau, \rho_1, \rho_2, s)$ and compute

$$\mathsf{com} := (C_0, C_1, C_2), \text{ for } \begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} g^{\tau} \\ X_1^{\rho_1} \cdot g^s \\ X_2^{\rho_2} \cdot h^s \end{pmatrix}.$$

- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$: Set $R := (R_1, R_2) := \left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right)$. Then, if $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute

$$\varphi := (\alpha, \beta, \gamma), \text{ for } \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} = \mathbf{D}^{-1} \cdot \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix}.$$

Theorem 4.3. *If the* DDH *assumption holds relative to* GGen*, then* $\mathsf{CMT}_{\mathsf{DDH}}$ *is a* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with*

$$\varepsilon_{\mathsf{b}} \leq 1/p, \quad \varepsilon_{\mathsf{g}} \leq 2/p, \quad \varepsilon_{\mathsf{t}} \leq 6/p.$$

Concretely, for any PPT algorithm $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

The homomorphism property is trivial to check. Next, we define the set Good as in the definition of a weakly equivocable commitment scheme. Namely, we define

$$\mathsf{Good} = \{((g,h),x) \in \mathbb{G}^2 \times \mathbb{Z}_p \mid (g,h) \in \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda) \wedge h \neq g^0 \wedge x \neq 0\}.$$

Clearly, for $(g,h) \leftarrow \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda)$ and $x \xleftarrow{\$} \mathbb{Z}_p$, the probability that $((g,h),x) \notin \mathsf{Good}$ is at most $2/p$. Therefore, $\varepsilon_{\mathsf{g}} \leq 2/p$. In the following we also need the following observation: If $((g,h),x) \in \mathsf{Good}$, then the elements g, h, g^x, h^x are all generators of $\mathbb{G}$. The rest of proof of the theorem is given in separate lemmas.

Lemma 4.6. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the uniform keys property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$.

Proof. Let $(\mathsf{par}, x) \in \mathsf{Good}$ for $\mathsf{par} = (g,h)$. Let $(X_1, X_2) = \mathsf{F}(x) = (g^x, h^x)$. Consider the distribution of ck for $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. Then ck has the form

$$\begin{pmatrix} g^{d_{1,1}} & g^{d_{1,2}} & g^{d_{1,3}} \\ X_1^{d_{2,1}} & X_1^{d_{2,2}} & X_1^{d_{2,3}} \\ X_2^{d_{3,1}} & X_2^{d_{3,2}} & X_2^{d_{3,3}} \end{pmatrix} \in \mathbb{G}^{3\times 3}$$

for uniformly random and independent exponents $d_{i,j} \in \mathbb{Z}_p$ $(i,j \in [3])$. As g, X_1, X_2 are generators, we see that ck is uniform over $\mathbb{G}^{3\times 3}$, proving the claim. □

Lemma 4.7. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the weak trapdoor property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, where* $\varepsilon_{\mathsf{t}} \leq 6/p$.

Proof. Let $((g,h),x) \in \mathsf{Good}$ and $c \in \mathbb{Z}_p$. Set $(X_1, X_2) := (g^x, h^x)$. We have to show that the distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ of tuples

$$((g,h), \mathbf{A}, \mathbf{D}, X_1, X_2, x, c, (C_0, C_1, C_2), \alpha, \beta, \gamma, R_1, R_2, s)$$

are identical. Here, we have $(\mathbf{A}, \mathbf{D}, X_1, X_2) \leftarrow \mathsf{TGen}(\mathsf{par}, (X_1, X_2))$. The remaining components in $\mathcal{T}_0$ are generated via

$$((C_0, C_1, C_2), St) \leftarrow \mathsf{TCom}(\mathsf{ck}, \mathsf{td}),\ ((\alpha, \beta, \gamma), (R_1, R_2), s) \leftarrow \mathsf{TCol}(St, c),$$

and in $\mathcal{T}_1$ via

$$\begin{aligned} r \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^r,\ R_2 := h^r,\ s := c \cdot x + r \\ \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)). \end{aligned}$$

First, we make the assumption that in both distributions, the matrix $\mathbf{D}$ has full rank. The probability that this does not hold can easily be bounded by $3/p$.

We can equivalently[9] write $\mathcal{T}_1$ as

$$\begin{aligned} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \alpha, \beta, \gamma \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma)). \end{aligned}$$

Using that $\mathbf{D}$ is full rank and g, X_1, X_2 are generators of $\mathbb{G}$, we see that in this distribution, (C_0, C_1, C_2) is uniform over $\mathbb{G}^3$. Therefore, this is identically distributed to the distribution that we get from

$$\begin{aligned} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \tau, \rho_1, \rho_2 \xleftarrow{\$} \mathbb{Z}_p,\ (C_0, C_1, C_2) := (g^\tau, X_1^{\rho_1} g^s, X_2^{\rho_2} h^s), \end{aligned}$$

[9] This corresponds to the HVZK property of linear identification protocols.

and then finding the unique values (α, β, γ) that satisfy $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$. We claim that this can be done using $(\alpha, \beta, \gamma)^t := \mathbf{D}^{-1}(\tau, \rho_1 + c, \rho_2 + c)^t$, which is equivalent to distribution $\mathcal{T}_0$.

To see this, note that $(C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (\alpha, \beta, \gamma))$ is equivalent to

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \cdot A_{1,3}^{\gamma} \\ R_1 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \cdot A_{2,3}^{\gamma} \\ R_1 \cdot & A_{3,1}^{\alpha} \cdot A_{3,2}^{\beta} \cdot A_{3,3}^{\gamma} \end{pmatrix} = \begin{pmatrix} & g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ g^s \cdot X_1^{-c} \cdot & X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ h^s \cdot X_2^{-c} \cdot & X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix}.$$

Using the way we generate (C_0, C_1, C_2), we see that the g^s and h^s terms cancel out, and this is equivalent to

$$\begin{pmatrix} g^{\tau} \\ X_1^{\rho_1} \\ X_2^{\rho_2} \end{pmatrix} = \begin{pmatrix} g^{d_{1,1}\alpha} \cdot g^{d_{1,2}\beta} \cdot g^{d_{1,3}\gamma} \\ X_1^{d_{2,1}\alpha} \cdot X_1^{d_{2,2}\beta} \cdot X_1^{d_{2,3}\gamma} \\ X_2^{d_{3,1}\alpha} \cdot X_2^{d_{3,2}\beta} \cdot X_2^{d_{3,3}\gamma} \end{pmatrix} \iff \begin{pmatrix} \tau \\ \rho_1 + c \\ \rho_2 + c \end{pmatrix} = \mathbf{D} \cdot \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix}.$$

This concludes the proof. □

Lemma 4.8. $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the statistically binding property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with* $\varepsilon_{\mathsf{b}} \leq 1/p$.

Proof. We describe an unbounded algorithm Ext, that takes as input a commitment key $\mathsf{ck} = \mathbf{A} = (A_{i,j})_{i,j} \in \mathbb{G}^{3\times 3}$, and a commitment $\mathsf{com} = (C_0, C_1, C_2) \in \mathbb{G}^3$, and outputs a tuple $R = (R_1, R_2) \in \mathbb{G} \times \mathbb{G}$. It is given as follows:

1. Extract discrete logarithms $\mathbf{c} = (c_0, c_1, c_2)^t \in \mathbb{Z}_p^3$ and $\mathbf{a} = (a_0, a_1, a_2)^t \in \mathbb{Z}_p^3$ such that
$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} g^{c_0} \\ g^{c_1} \\ g^{c_2} \end{pmatrix} \text{ and } \begin{pmatrix} A_{1,1} \\ A_{2,1} \\ A_{3,1} \end{pmatrix} = \begin{pmatrix} g^{a_0} \\ g^{a_1} \\ g^{a_2} \end{pmatrix}.$$
2. If $a_0 = 0$, return $\perp$. Otherwise, let $\mathbf{e}_2 = (0, 1, 0)^t$ and $\mathbf{e}_3 = (0, 0, 1)^t$. Note that $\mathbf{a}, \mathbf{e}_2, \mathbf{e}_3$ form a basis of $\mathbb{Z}_p^3$.
3. Write $\mathbf{c}$ as $\mathbf{c} = t\mathbf{a} + r_1\mathbf{e}_2 + r_2\mathbf{e}_3$ for $t, r_1, r_2 \in \mathbb{Z}_p$, and return $(R_1, R_2) := (g^{r_1}, g^{r_2})$.

To finish the proof, let $\mathcal{A}$ be any algorithm. We have to bound the probability

$$\Pr\left[\begin{array}{ll} \mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') & = (C_0, C_1, C_2) \\ \wedge\ (R_1, R_2) & \neq (R_1', R_2') \end{array} \;\middle|\; \begin{array}{l} (g, h) \leftarrow \mathsf{LF.Gen}(1^{\lambda}), \\ \mathbf{A} \leftarrow \mathsf{BGen}(\mathsf{par}), \\ ((C_0, C_1, C_2), St) \leftarrow \mathcal{A}(\mathbf{A}), \\ (R_1, R_2) \leftarrow \mathsf{Ext}(\mathbf{A}, (C_0, C_1, C_2)), \\ (R_1, R_2', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

Note that the probability that Ext outputs $\perp$ in this experiment is $1/p$, as $A_{1,1}$ is uniform in $\mathbb{G}$. We assume that Ext does not output $\perp$, and want to show that the above probability conditioned on this event is zero. First, it is easy to see that we have $\mathsf{Com}(\mathbf{A}, (R_1, R_2); (t, 0, 0)) = (C_0, C_1, C_2)$. Further, assume that $\mathcal{A}$ outputs $(R_1', R_2') = (g^{r_1'}, g^{r_2'})$ and $\varphi' = (\alpha, \beta, \gamma)$ such that

$$\mathsf{Com}(\mathbf{A}, (R_1', R_2'); \varphi') = (C_0, C_1, C_2) = \mathsf{Com}(\mathbf{A}, (R_1, R_2); (t, 0, 0)).$$

Using the definition of Com and BGen, we see that this implies the vector $(0, r_1 - r_1', r_2 - r_2')^t$ is in the span of $\mathbf{a}$. As $a_0 \neq 0$ this implies that it is the zero vector, showing that $R_1 = R_1'$ and $R_2 = R_2'$. □

Lemma 4.9. *For any PPT algorithm $\mathcal{A}$, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{uDDH3}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{6}{p}.$$

Proof. Let $\mathcal{A}$ be a PPT algorithm and $Q = \mathsf{poly}(\lambda)$. We have to bound

$$\left|\Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right] - \Pr\left[Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \Rightarrow 1\right]\right|.$$

Note that in game $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$, all commitment keys $\mathbf{A}_i$ are sampled uniformly at random. Therefore, looking at one fixed commitment key $\mathbf{A}_i$, indistinguishability would directly follow from the uDDH3 assumption. To give a tight reduction for any $Q = \mathsf{poly}(\lambda)$, we use the random self-reducibility of uDDH3. Our reduction $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ gets as input $\mathbb{G}, p, g$ and group elements $(h_{i,j})_{i,j\in[3]}$.
2. $\mathcal{B}$ samples $h \xleftarrow{\$} \mathbb{G}$, sets $\mathsf{par} := (g, h)$, and samples $x \xleftarrow{\$} \mathbb{Z}_p$.
3. If $(\mathsf{par}, x) \notin \mathsf{Good}$, $\mathcal{B}$ returns 0, as game $\mathbf{KEYDIST}_{b,\mathsf{CMT}_{\mathsf{DDH}}}$ does.
4. Otherwise, $\mathcal{B}$ prepares commitment keys $\mathsf{ck}_i := \mathbf{A}_i \in \mathbb{G}^{3\times 3}$ for all $i \in [Q]$ as follows:
   (a) For simplicity of notation, write $h_{i,j} := g^{\mathbf{H}_{i,j}}$, i.e., let $\mathbf{H} \in \mathbb{Z}_p^{3\times 3}$ denote the matrix of exponents of $h_{i,j}$. Partition $\mathbf{H}$ into $\mathbf{H} = [\mathbf{H}_0 \mid \mathbf{H}_1]$ for $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ and $\mathbf{H}_1 \in \mathbb{Z}_p^{3\times 2}$.
   (b) Reduction $\mathcal{B}$ samples $\mathbf{T}_i \xleftarrow{\$} \mathbb{Z}_p^{3\times 3}$, and $\mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_p^{1\times 2}$. We define
   $$\mathbf{D}_i := \mathbf{T}_i\mathbf{H} + [\mathbf{0} \mid \mathbf{T}_i\mathbf{H}_0\mathbf{S}_i].$$
   (c) Reduction $\mathcal{B}$ computes $\mathbf{A}_i := g^{\mathbf{D}_i}$, which should be understood componentwise. It is easy to see that $\mathcal{B}$ can efficiently compute $\mathbf{A}_i$, given $\mathbf{T}_i$, $\mathbf{S}_i$, and $(h_{i,j})_{i,j\in[3]}$.
5. $\mathcal{B}$ runs $\beta \leftarrow \mathcal{A}(\mathsf{par}, x, (\mathsf{ck}_i)_{i\in[Q]})$ and returns β.

Apart from the distribution of the commitment keys $\mathbf{A}_i$, it is clear that $\mathcal{B}$ simulates the games perfectly for $\mathcal{A}$. We claim that if the $h_{i,j}$ are uniform and independent, then $\mathcal{B}$ provides a simulation statistically close to $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$, i.e., all $\mathbf{A}_i$ are uniform and independent. If on the other hand, there are a, b such that $h_{i,2} := h_{i,1}^a, h_{i,3} := h_{i,1}^b$ for all $i \in [3]$, then $\mathcal{B}$ does the same for $\mathbf{KEYDIST}_{1,\mathsf{CMT}_{\mathsf{DDH}}}$.

For the first claim, assume that the $h_{i,j}$ are uniform and independent, i.e., $\mathbf{H}$ is uniform over $\mathbb{Z}_p^{3\times 3}$. Then with probability at least $1 - 3/p$ it has full rank, i.e., is invertible. Assuming that it has full rank, we see that even if we fix the matrix $\mathbf{H}$, the matrix $\mathbf{T}_i\mathbf{H}$ is uniform over $\mathbb{Z}_p^{3\times 3}$. This implies that $\mathbf{D}_i$ is uniform, showing the first claim.

For the second claim, assume that there are a, b such that $h_{i,2} := h_{i,1}^a, h_{i,3} := h_{i,1}^b$ for all $i \in [3]$. This is equivalent to writing $\mathbf{H} = [\mathbf{H}_0 \mid \mathbf{H}_0\mathbf{R}]$, where $\mathbf{R} = [a, b] \in \mathbb{Z}_p^{1\times 2}$. With probability at least $1 - 1/p^3$, the matrix $\mathbf{H}_0 \in \mathbb{Z}_p^{3\times 1}$ is full rank. We see that

$$\mathbf{D}_i = \mathbf{T}_i\mathbf{H} + [\mathbf{0} \mid \mathbf{T}_i\mathbf{H}_0\mathbf{S}_i] = [\mathbf{T}_i\mathbf{H}_0 \mid \mathbf{T}_i\mathbf{H}_0(\mathbf{R} + \mathbf{S})].$$

If $\mathbf{H}_0$ has full rank, we see that (even for fixed $\mathbf{H}$ of this form) the key $\mathbf{A}_i$ is distributed exactly as a commitment key in $\mathbf{KEYDIST}_{0,\mathsf{CMT}_{\mathsf{DDH}}}$, which finishes the proof. □

4.5 Toothpicks: Reducing the Price of Tightness

The Chopsticks schemes presented in Section 4.4 are the first two-round multi-signature schemes in the pairing-free discrete logarithm setting that do not rely on rewinding. However, the price we pay for this improvement is a large signature and communication size. In this section, we formally present and analyze our constructions of two-round multi-signatures from [PW24]: rewinding-free schemes without such an efficiency penalty. We follow the structure of the previous section. That is, we first define the abstract building blocks we need and then show two abstract constructions $\mathsf{ToothKA}[\mathsf{LF}, \mathsf{CMT}]$ and $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$ using these building blocks. Finally, we instantiate LF and CMT, obtaining the schemes Toothpicks I and Toothpicks II, respectively, and discuss minor optimizations and the concrete efficiency of our schemes.

4.5.1 Building Blocks: Stronger Linear Functions and Weaker Commitments

We introduce two building blocks we will use in our constructions. As in Section 4.4, we make use of linear function families and a special kind of commitment scheme. However, compared to Section 4.4, a crucial observation is that we can weaken the requirements for the commitment scheme, thereby enabling a more efficient instantiation. To compensate, we strengthen the requirements for the linear function family in terms of soundness, which is for free in terms of efficiency.

Stronger Linear Function Families. In Section 4.4.1, we have defined linear function families. We have also defined key indistinguishability and lossy soundness to capture the set of properties that makes linear function families amenable for the use in lossy identification [AFLT12]. Here, we will strengthen the definition of lossy soundness, which will allow us to weaken the properties for commitments. Concretely, we relax the winning condition, such that it has to hold up to an arbitrary shift in the image of the linear function, leading to coset lossy soundness. We also adapt the definition of aggregation lossy soundness accordingly, leading to coset aggregation lossy soundness. Importantly, we show in Lemmas 4.10 and 4.11 that the strengthened definitions come for free.

Definition 4.9 (Coset Lossy Soundness)**.** *Let* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *be a linear function family. We say that* LF *satisfies* ε_{l}*-coset lossy soundness, if for any unbounded algorithm* $\mathcal{A}$*, the following probability is at most* ε_{l}*:*

$$\Pr\left[\mathsf{F}(s) - c \cdot X \in R + \mathsf{F}(\mathcal{D}) \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X \xleftarrow{\$} \mathcal{R}, \\ (St, R) \leftarrow \mathcal{A}(\mathsf{par}, X), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St, c) \end{array}\right].$$

Lemma 4.10. *Let* LF *be a linear function family, such that for any* $\mathsf{par} \in \mathsf{Gen}(1^\lambda)$*, the domain* $\mathcal{D}_{\mathsf{par}}$ *can be enumerated. Then, if* LF *satisfies* ε_{l}*-lossy soundness, it also satisfies* ε_{l}*-coset lossy soundness.*

Proof. To prove the claim, it is sufficient to describe an (unbounded) reduction $\mathcal{B}$, that turns any algorithm $\mathcal{A}$ running in the coset lossy soundness game into an algorithm in the lossy soundness game. The reduction $\mathcal{B}$ gets as input parameters par and an element $X \in \mathcal{R}$ from the lossy soundness game. It runs $\mathcal{A}$ on input par and X and gets an output R in return, which it passes to the lossy soundness game. In return, it receives $c \in \mathcal{S}$, and forwards it to $\mathcal{A}$, which outputs $s \in \mathcal{D}$. If $\mathcal{A}$ breaks coset lossy soundness, i.e, $\mathsf{F}(s) - c \cdot X \in R + \mathsf{F}(\mathcal{D})$, then there is a $\delta \in \mathcal{D}$ such that $\mathsf{F}(s) - c \cdot X = R + \mathsf{F}(\delta)$. By enumerating $\mathcal{D}$, the reduction $\mathcal{B}$ finds such a δ and returns $s - \delta$ to the lossy soundness game. As we have $\mathsf{F}(s - \delta) - c \cdot X = R$, $\mathcal{B}$ breaks lossy soundness with the same probability as $\mathcal{A}$ breaks coset lossy soundness, and the claim follows. □

Definition 4.10 (Coset Aggregation Lossy Soundness)**.** *Let* $\mathsf{LF} = (\mathsf{Gen}, \mathsf{F})$ *be a linear function family. We say that* LF *satisfies* $\varepsilon_{\mathsf{al}}$*-coset aggregation lossy soundness, if for any unbounded algorithm* $\mathcal{A}$*, the*

following probability is at most $\varepsilon_{\mathsf{al}}$*:*

$$\Pr\left[\mathsf{F}(s) - c\cdot\tilde{X} \in R + \mathsf{F}(\mathcal{D}) \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ X_1 \xleftarrow{\$} \mathcal{R}, \\ (St, (X_2, a_2), \ldots, (X_N, a_N)) \leftarrow \mathcal{A}(\mathsf{par}, X_1), \\ a_1 \xleftarrow{\$} \mathcal{S},\ (St', R) \leftarrow \mathcal{A}(St, a_1), \\ c \xleftarrow{\$} \mathcal{S},\ s \leftarrow \mathcal{A}(St', c), \\ \tilde{X} := \sum_{i=1}^{N} a_i X_i \end{array}\right].$$

Lemma 4.11. *Let* LF *be a linear function family, such that for any* $\mathsf{par} \in \mathsf{Gen}(1^\lambda)$*, the domain* $\mathcal{D}_{\mathsf{par}}$ *can be enumerated. Then, if* LF *satisfies* $\varepsilon_{\mathsf{al}}$*-aggregation lossy soundness, it also satisfies* ε_{l}*-coset aggregation lossy soundness.*

Proof. The proof is almost identical to the proof of Lemma 4.10, and details are left to the reader. □

Weaker Commitments. Next, we weaken the commitment definition given in Section 4.4.1. To recall, a weakly equivocable commitment scheme as defined in Section 4.4.1 allows to homomorphically commit to elements in the range of a linear function family. In addition to a statistically binding mode, there is an indistinguishable way of generating commitment keys together with a weak equivocation trapdoor. This trapdoor allows to open commitments to all messages of a certain structure. In comparison to Section 4.4.1, we now weaken the binding property of the scheme. Concretely, in the binding mode, we only require the commitment to be binding up to any shift in the image of the linear function.

Definition 4.11 (Weakly Equivocable Coset Commitment Scheme)**.** *Let* $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ *be a linear function family and* $\mathcal{G} = \{\mathcal{G}_{\mathsf{par}}\}, \mathcal{H} = \{\mathcal{H}_{\mathsf{par}}\}$ *be families of subsets of abelian groups with efficiently computable group operations* $\oplus$ *and* $\otimes$*, respectively. Let* $\mathcal{K} = \{\mathcal{K}_{\mathsf{par}}\}$ *be a family of sets. An* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* LF *with key space* $\mathcal{K}$*, randomness space* $\mathcal{G}$ *and commitment space* $\mathcal{H}$ *is a tuple of PPT algorithms* $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ *with the same syntax as a weakly equivocable commitment scheme according to Definition 4.8. Further, the algorithms are required to satisfy the following properties:*

- ***Homomorphism.*** *As in Definition 4.8.*
- ***Good Parameters.*** *As in Definition 4.8.*
- ***Uniform Keys.*** *As in Definition 4.8.*
- ***Weak Trapdoor Property.*** *As in Definition 4.8.*
- ***Multi-Key Indistinguishability.*** *As in Definition 4.8.*
- ***Statistical Coset Binding.*** *There exists some (potentially unbounded) algorithm* Ext*, such that for every (potentially unbounded) algorithm* $\mathcal{A}$ *the following probability is at most* ε_{b}*:*

$$\Pr\left[\mathsf{Com}(\mathsf{ck}, R'; \varphi') = \mathsf{com} \wedge R' \notin R + \mathsf{F}(\mathcal{D}) \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda), \\ \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par}),\ (\mathsf{com}, St) \leftarrow \mathcal{A}(\mathsf{ck}), \\ R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com}),\ (R', \varphi') \leftarrow \mathcal{A}(St) \end{array}\right].$$

4.5.2 Construction with Key Aggregation

In Section 4.4.2, we have used a linear function family LF and a weakly equivocable commitment scheme CMT to construct a multi-signature scheme $\mathsf{ChopsKA}[\mathsf{LF}, \mathsf{CMT}]$ supporting key aggregation. For the security proof to work, LF has to satisfy aggregation lossy soundness. Here, we show that if CMT is a weakly equivocable coset commitment scheme and LF satisfies coset aggregation lossy soundness, then the very same construction is also secure, with the same security loss. To distinguish these two ways of instantiating the scheme, we write $\mathsf{ToothKA}[\mathsf{LF}, \mathsf{CMT}]$ for the new scheme. For a description of the scheme, the reader may consult Section 4.4.2 and Figure A.1. We now state completeness, which is easy to verify, and prove security.

Lemma 4.12. *Let* LF *be a linear function family. Let* CMT *be an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* LF *Then* $\mathsf{ToothKA}[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

Theorem 4.4. *Let* LF *be a linear function family that satisfies key indistinguishability and* $\varepsilon_{\mathsf{al}}$*-coset aggregation lossy soundness. Let* CMT *be an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$*,* $\mathsf{H}_a\colon \{0,1\}^* \to \mathcal{S}$*, and* $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then, the scheme* $\mathsf{ToothKA}[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}}, Q_{\mathsf{H}_a}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_a, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{ToothKA}[\mathsf{LF},\mathsf{CMT}]}(\lambda) \leq\ & \varepsilon_{\mathsf{g}} + 4Q_S^2\varepsilon_{\mathsf{t}} + 4Q_S\varepsilon_{\mathsf{g}} + 4Q_S Q_{\mathsf{H}} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} \\ &+ \frac{4Q_S}{|\mathcal{R}|} + \frac{4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|} + 4Q_S Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}\varepsilon_{\mathsf{al}} \\ &+ 4Q_S\left(\mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda)\right).\end{aligned}$$

Proof. The proof is an adaptation of the proof of Theorem 4.1 to our new building blocks. Therefore, we only sketch the proof informally. The proof is identical to the proof of Theorem 4.1 except for $\mathbf{G}_7$ and the fact that the final reduction reduces to coset aggregation lossy soundness instead of aggregation lossy soundness.

Game $\mathbf{G}_0$: This is the original security game $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{ToothKA}[\mathsf{LF},\mathsf{CMT}]}$. As in the proof of Theorem 4.1, we can omit signing oracle $\textsc{Sig}_2$ without loss of generality. We have

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{ToothKA}[\mathsf{LF},\mathsf{CMT}]}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_1$: We let the game abort if $(\mathsf{par}, x_1) \notin \mathsf{Good}$, where x_1 is the secret key of the signer that is simulated by the game. As in the proof of Theorem 4.1, we can argue that

$$\left|\Pr\left[\mathbf{G}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1 \Rightarrow 1\right]\right| \leq \Pr\left[(\mathsf{par}, x_1) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_2$: We introduce a map b mapping inputs $(\tilde{\mathsf{pk}}, \mathsf{m})$ to random oracle H to bits. For each such input for which $b[\tilde{\mathsf{pk}}, \mathsf{m}]$ is not yet defined, we set it to 1 with probability $1/(Q_S+1)$, and to 0 otherwise. The game additionally aborts if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$ for a signing query or $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$ for the forgery. We can argue that

$$\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] \geq \frac{1}{4Q_S} \cdot \Pr\left[\mathbf{G}_1 \Rightarrow 1\right].$$

Game $\mathbf{G}_3$: We change how commitment keys ck output by random oracle H on inputs $(\tilde{\mathsf{pk}}, \mathsf{m})$ are sampled. Namely, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$, then we sample ck with a trapdoor via $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_1)$, where X_1 is the public key of the signer simulated by the game. Otherwise, if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$, we sample ck in binding mode via $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. Indistinguishability can be argued using the uniform keys property of CMT and the multi-key indistinguishability property of CMT. We get a reduction $\mathcal{B}$ with

$$\left|\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_3 \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_4$: In this game, we use the trapdoor td generated in random oracle H to simulate the signing oracle. This works out because due to the changes in $\mathbf{G}_2$ and $\mathbf{G}_3$, the commitment key ck used in signing queries has been sampled with a trapdoor. We can argue that

$$\left|\Pr\left[\mathbf{G}_3 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_4 \Rightarrow 1\right]\right| \leq Q_S\varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_5$: We undo the change from $\mathbf{G}_1$ and no longer require that $(\mathsf{par}, x_1) \in \mathsf{Good}$. As before, we get

$$\left|\Pr\left[\mathbf{G}_4 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_5 \Rightarrow 1\right]\right| \leq \Pr\left[(\mathsf{par}, x_1) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_6$: We change how public key X_1 is generated. Before, it was generated by sampling the secret key $x_1 \xleftarrow{\$} \mathcal{D}$ and setting $X_1 := \mathsf{F}(x_1)$. Now, we sample $X_1 \xleftarrow{\$} \mathcal{R}$. Note that the secret key x_1 is not used anymore, due to the previous changes. Now, we can use the key indistinguishability of LF and get a reduction $\mathcal{B}'$ with

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_7$: We use the statistical coset binding property of CMT as follows. For oracle queries of the form $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$, we set $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$, and extract from the commitment com using the extractor Ext from the statistical coset binding property of CMT if ck has been sampled in binding mode, i.e., if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$. Concretely, we run $R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ and store R in another map r as $r[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := R$. We continue the simulation of H_c as before. Later, when $\mathcal{A}$ outputs its forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ for a signature $\sigma^* = (\mathsf{com}^*, s^*, \varphi^*)$, we compute the aggregated key $\tilde{\mathsf{pk}} := \tilde{X} := \mathsf{Agg}(\mathcal{P}^*)$ and $c^* := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*)$ and $R^* := \mathsf{F}(s^*) - c^* \cdot \tilde{X}$ as in the verification algorithm. Then, the game outputs 0 if $R^* \notin r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*] + \mathsf{F}(\mathcal{D})$. Otherwise, it continues as before. Observe that compared to $\mathbf{G}_7$ in the proof of Theorem 4.1, we weaken this new winning condition by allowing a difference in the span of F. This is necessary as we only have statistical coset binding, and not statistical binding as in Theorem 4.1. We will see that this is compensated by coset aggregation lossy soundness. Using a reduction as given in $\mathbf{G}_7$ in the proof of Theorem 4.1, we can argue that

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

Game $\mathbf{G}_8$: We ensure that H_a and H_c are always queried in the correct order. Namely, if there is a query $\mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk})$ for $\mathsf{pk} = X_1$ and the hash value is not yet defined, but for $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$ the hash value $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$ is already defined for some com, m, then the game aborts. As in $\mathbf{G}_8$ in the proof of Theorem 4.1, we get that

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq \frac{1}{|\mathcal{R}|} + \frac{Q_{\mathsf{H}_a} Q_{\mathsf{H}_c}}{|\mathcal{S}|}.$$

Finally, we reduce from coset aggregation lossy soundness to finish the proof. In contrast to the final reduction in the proof of Theorem 4.1, it is important to use coset aggregation lossy soundness and not aggregation lossy soundness due to the modified change in $\mathbf{G}_7$. We get

$$\Pr[\mathbf{G}_8 \Rightarrow 1] \leq Q_{\mathsf{H}_a} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{al}}.$$

□

4.5.3 Tight Construction

Here, we present our construction of a tightly secure two-round multi-signature scheme. In comparison to our first tightly secure construction in Section 4.4.3, this construction is significantly more efficient. Let $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ be a linear function family. Let $\mathsf{CMT} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ be an $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$-weakly equivocable coset commitment scheme for LF with key space $\mathcal{K}$, randomness space $\mathcal{G}$ and commitment space $\mathcal{H}$. Finally, let $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}$, $\mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}$, and $\mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ be random oracles. We give a verbal description of our scheme $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. In addition, we present it as pseudocode in Figure A.8.

Setup and Key Generation. Our scheme makes use of public parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$, which define the linear function $\mathsf{F} = \mathsf{F}(\mathsf{par}, \cdot)$. Keys are generated by sampling elements $x_0, x_1 \xleftarrow{\$} \mathcal{D}$ and a seed $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$. Then, the keys are

$$\mathsf{sk} := (x_0, x_1, \mathsf{seed}), \quad \mathsf{pk} := (X_0, X_1) := (\mathsf{F}(x_0), \mathsf{F}(x_1)).$$

Signing Protocol. We consider the setting of a set of N signers with public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$. Let $\mathsf{m} \in \{0,1\}^*$ denote the message that should be signed. In the following, we describe the signing

protocol, i.e., algorithms $\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2$, from the perspective of the first signer. This signer holds a secret key $\mathsf{sk}_1 = (x_{1,0}, x_{1,1}, \mathsf{seed}_1)$ for public key $\mathsf{pk}_1 = (X_{1,0}, X_{1,1})$.

1. *Commitment Phase.* First, a commitment key $\mathsf{ck} := \mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$ is derived from the set of public keys and the message. Further, the signer computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$. The signer computes
$$r_1 \xleftarrow{\$} \mathcal{D}, \;\; R_1 := \mathsf{F}(r_1).$$
Then, the signer commits to R_1 using the commitment key ck, i.e., it computes
$$\varphi_1 \xleftarrow{\$} \mathcal{G}, \;\; \mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1).$$
Finally, it sends $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_1)$ as its first message of the protocol to all signers.

2. *Response Phase.* Let $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N})$ be the list of messages output by the signers in the commitment phase. That is, the message $\mathsf{pm}_{1,i}$ is sent by signer i and has the form $\mathsf{pm}_{1,i} = (b_i, \mathsf{com}_i)$. The signer aggregates these messages by setting
$$B := b_1 \ldots b_N \in \{0,1\}^N, \;\; \mathsf{com} := \bigotimes_{i\in[N]} \mathsf{com}_i.$$
Next, a signer specific challenge c_1 is derived and a response s_1 is computed. This is done via
$$c_1 := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B), \;\; s_1 := c_1 \cdot x_{1,b_1} + r_1.$$
Observe that the signer uses bit b_1 to determine which part of the secret key is used. Finally, the signer sends $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$ as its second message of the protocol to all signers.

3. *Aggregation Phase.* Let $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N})$ be the list of messages output by the signers in the response phase. That is, the message $\mathsf{pm}_{2,i}$ is sent by signer i and has the form $\mathsf{pm}_{2,i} = (s_i, \varphi_i)$. The signers aggregate the responses and commitment randomness received in the previous messages via
$$s := \sum_{i\in[N]} s_i, \;\; \varphi := \bigoplus_{i\in[N]} \varphi_i.$$
Finally, the signature is defined as $\sigma := (\mathsf{com}, \varphi, s, B)$.

Verification. Assume we have a set of public keys $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, a message $\mathsf{m} \in \{0,1\}^*$, and a signature $\sigma := (\mathsf{com}, \varphi, s, B)$. To verify σ, write $B = b_1 \ldots b_N \in \{0,1\}^N$ and each public key pk_i as $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$. Then, reconstruct the commitment key $\mathsf{ck} := \mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$ and the signer specific challenges $c_i := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$ for each $i \in [N]$. The signature is valid, i.e., the verification outputs 1, if and only if
$$\mathsf{com} = \mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - \sum_{i=1}^{N} c_i \cdot X_{i,b_i}; \varphi\right).$$

Lemma 4.13. *Let* LF *be a linear function family. Let* CMT *be an* $(\varepsilon_\mathsf{b}, \varepsilon_\mathsf{g}, \varepsilon_\mathsf{t})$*-weakly equivocable coset commitment scheme for* LF*. Then* $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$ *is complete.*

Proof. To show completeness of $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$, consider N users and let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ be the set of their public keys, where $\mathsf{pk}_i = (X_{i,0}, X_{i,1}) = (\mathsf{F}(x_{i,0}), \mathsf{F}(x_{i,1}))$ for each $i \in [N]$. Let $\mathsf{m} \in \{0,1\}^*$ be a message, and let $\sigma = (\mathsf{com}, \varphi, s, B)$ for $B = b_1 \ldots b_N \in \{0,1\}^N$ be a signature computed honestly in the signing protocol. We have to show that verification outputs 1 on input

$\mathcal{P}, \mathsf{m}, \sigma$. For that, let $\mathsf{ck} = \mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$ and $c_i = \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$ for each $i \in [N]$ be as in the verification algorithm. We have to show that

$$\mathsf{com} = \mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - \sum_{i=1}^{N} c_i \cdot X_{i,b_i}; \varphi\right).$$

Using definition of s and the X_{i,b_i}, and linearity of F, we get

$$\mathsf{F}(s) - \sum_{i=1}^{N} c_i \cdot X_{i,b_i} = \mathsf{F}\left(\sum_{i=1}^{N} s_i\right) - \sum_{i=1}^{N} c_i \cdot \mathsf{F}(x_{i,b_i}) = \sum_{i=1}^{N} \mathsf{F}(s_i - c_i \cdot x_{i,b_i}).$$

Now, we use the definition of the s_i as $s_i = c_i \cdot x_{i,b_i} + r_i$, where $r_i \in \mathcal{D}$ is the element that the ith signer samples in the first step, and get

$$\sum_{i=1}^{N} \mathsf{F}(s_i - c_i \cdot x_{i,b_i}) = \sum_{i=1}^{N} \mathsf{F}(r_i) = \sum_{i=1}^{N} R_i,$$

where $R_i = \mathsf{F}(r_i)$ is the element to which each signer commits in the first step. In combination, we get

$$\mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - \sum_{i=1}^{N} c_i \cdot X_{i,b_i}; \varphi\right) = \mathsf{Com}\left(\mathsf{ck}, \sum_{i=1}^{N} R_i; \bigoplus_{i\in[N]} \varphi_i\right),$$

where we used the definition of φ. We can now apply the homomorphism property of the commitment and get

$$\mathsf{Com}\left(\mathsf{ck}, \sum_{i=1}^{N} R_i; \bigoplus_{i\in[N]} \varphi_i\right) = \bigotimes_{i=1}^{N} \mathsf{Com}\left(\mathsf{ck}, R_i; \varphi_i\right) = \bigotimes_{i=1}^{N} \mathsf{com}_i = \mathsf{com},$$

where the com_i are what each signer sends in the first message. This proves the claim. □

Theorem 4.5. *Let* LF *be a linear function family that satisfies key indistinguishability and* ε_{l}*-coset lossy soundness. Let* CMT *be an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* LF*. Further, let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{K}, \mathsf{H}_b\colon \{0,1\}^* \to \{0,1\}, \mathsf{H}_c\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Then* $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$ *is* $\mathsf{MS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* $Q_{\mathsf{H}}, Q_{\mathsf{H}_b}, Q_{\mathsf{H}_c}, Q_S$ *queries to oracles* $\mathsf{H}, \mathsf{H}_b, \mathsf{H}_c, \textsc{Sig}_0$*, respectively, there are PPT algorithms* $\mathcal{B}, \mathcal{B}'$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A}), \mathbf{T}(\mathcal{B}') \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Tooth}[\mathsf{LF},\mathsf{CMT}]}(\lambda) \leq \frac{Q_{\mathsf{H}_b}}{2^{\lambda}} + 8\varepsilon_{\mathsf{g}} + 4Q_S\varepsilon_{\mathsf{t}} + 4Q_{\mathsf{H}}Q_{\mathsf{H}_c}\varepsilon_{\mathsf{b}} + 4Q_{\mathsf{H}_c}\varepsilon_{\mathsf{l}} \\ + 4 \cdot \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda) + 4 \cdot \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).\end{aligned}$$

Proof. Let $\mathcal{A}$ be an adversary against the security of $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$. To prove the statement, we give a sequence of games $\mathbf{G}_0, \ldots, \mathbf{G}_8$. We present the games formally in Figures A.9 to A.11, and we verbally describe and analyze them here.

Game $\mathbf{G}_0$: This is defined to be the original security game $\mathbf{MS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{Tooth}[\mathsf{LF},\mathsf{CMT}]}$, but we omit the oracle $\textsc{Sig}_2$ from the game. Observe that this is without loss of generality for the scheme at hand, as this oracle can be run publicly based on the outputs of the other oracles and does not make use of any secret state or key. More concretely, for any adversary $\mathcal{A}$ that calls this oracle, we can build a wrapper adversary that internally simulates the game including oracle $\textsc{Sig}_2$ for $\mathcal{A}$ and forwards everything else to $\mathbf{G}_0$. This wrapper adversary has the same advantage and running time as $\mathcal{A}$. We now recall

the game to fix notation. First, system parameters $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$ are generated. In addition, the secret and public key of an honest user are generated. Namely, the game samples $\mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$ and $x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D}$ and sets $X_{1,0} := \mathsf{F}(x_{1,0})$ and $X_{1,1} := \mathsf{F}(x_{1,1})$. It sets $\mathsf{pk}^* := (X_{1,0}, X_{1,1})$ and runs $\mathcal{A}$ on input $\mathsf{par}, \mathsf{pk}^*$, with access to the following oracles

- Signing oracles $\textsc{Sig}_0$ and $\textsc{Sig}_1$: The signing oracles simulate an honest signer in a signing interaction. More precisely, if $\mathcal{A}$ queries $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$, a new signing interaction for message m with respect to $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ is started, where we assume that $\mathsf{pk}_1 = \mathsf{pk}^*$. For that, first $(\mathcal{P}, \mathsf{m})$ is added to list $\mathsf{Queried}$. Then, the game runs algorithm Sig_0 in the natural way and outputs the result to the adversary. Similarly, when $\mathcal{A}$ calls $\textsc{Sig}_1$, algorithm Sig_1 is run.
- Random oracles H and H_c: The game simulates random oracles H and H_c for $\mathcal{A}$ by standard lazy sampling. For that, it holds maps h and h_c which map the inputs to their outputs. For example, if $\mathcal{A}$ queries $\mathsf{H}(x)$, the game checks if $h[x]$ is defined. If it is not yet defined, it is sampled at random from the output domain of H, i.e., from $\mathcal{K}$. Then, the game returns $h[x]$.
- Random oracle H_b: For H_b, we additionally introduce a level of indirection. This will allow us to distinguish queries to H_b that the game itself issues from the queries that $\mathcal{A}$ issues directly. Concretely, when H_b is queried, the game forwards the query to a random oracle $\bar{\mathsf{H}}_b$ with the same interface. Oracle $\bar{\mathsf{H}}_b$ is simulated using a map $\bar{h}_b$ via lazy sampling. We emphasize that this oracle $\bar{\mathsf{H}}_b$ is not provided to $\mathcal{A}$. Further, the convention for all games will be that the game itself only queries $\bar{\mathsf{H}}_b$ and not H_b, for example in oracle $\textsc{Sig}_0$.

Finally, $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$. Write the set $\mathcal{P}^*$ as $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ and the signature σ^* as $\sigma^* = (\mathsf{com}^*, \varphi^*, s^*, B^*)$. Further, write $B^* = b_1^* \ldots b_N^* \in \{0,1\}^N$. Then, the game outputs 0 if $\mathsf{pk}^* \notin \mathcal{P}^*$ or $(\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$. Otherwise, we assume that $\mathsf{pk}^* = \mathsf{pk}_1$, and the game outputs 1 if and only if $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) = 1$. By definition, we have

$$\mathsf{Adv}^{\mathsf{MS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Tooth}[\mathsf{LF},\mathsf{CMT}]}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Before we continue, we give an overview of the remaining games and our strategy. In our first step (games $\mathbf{G}_1$ and $\mathbf{G}_2$), we ensure that for the forgery it holds that $b_1^* = 1 - b^*$ and $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = b^*$, for a random bit b^*. Once this is established, we change how we simulate the signing oracles (games $\mathbf{G}_3$ to $\mathbf{G}_6$). Namely, in the case $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m}) = b^*$, we embed a binding commitment key and simulate signing for $(\mathcal{P}, \mathsf{m})$ honestly, whereas for the other case, we embed a commitment key with a trapdoor and simulate signing by using the trapdoor. The result is that we no longer need $x_{1,1-b^*}$. Now, we switch $X_{1,1-b^*}$ to lossy mode and use the binding property to reduce to lossy soundness (games $\mathbf{G}_7$ and $\mathbf{G}_8$). This works, because the forgery is with respect to a lossy key and a binding commitment key.

Game $\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$, but we introduce a bad event on which the game aborts. Namely, the game sets $\mathsf{bad} := 1$ if $\mathcal{A}$ queries $\mathsf{H}_b(\mathsf{seed}_1, x)$ for any $x \in \{0,1\}^*$. Once $\mathcal{A}$ terminates, the game outputs 0 if $\mathsf{bad} = 1$. Otherwise, it behaves as $\mathbf{G}_0$. It is clear that games $\mathbf{G}_0$ and $\mathbf{G}_1$ only differ if $\mathcal{A}$ makes such a query. Further, the only information about seed_1 that $\mathcal{A}$ gets are the values of $\mathsf{H}_b(\mathsf{seed}_1, \cdot)$. As seed_1 is sampled uniformly at random from $\{0,1\}^\lambda$, we can bound the probability that a fixed query of $\mathcal{A}$ has the form $\mathsf{H}_b(\mathsf{seed}_1, x)$ by $1/2^\lambda$. With a union bound over the queries of $\mathcal{A}$ we obtain

$$\left|\Pr\left[\mathbf{G}_0 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_1 \Rightarrow 1\right]\right| \leq \frac{Q_{\mathsf{H}_b}}{2^\lambda}.$$

Game $\mathbf{G}_2$: In this game, we introduce a random bit $b^* \xleftarrow{\$} \{0,1\}$ that is sampled at the beginning of the game. Further, we change the winning condition as follows. When $\mathcal{A}$ outputs the forgery, the game outputs 0, if $b_1^* = b^*$ or $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = 1 - b^*$. Otherwise, it continues as $\mathbf{G}_1$ does. In other words, game $\mathbf{G}_2$ outputs 1 if $\mathbf{G}_1$ outputs 1 and the following event occurs:

- Event RightBits: This event occurs, if for $\mathcal{A}$'s final output $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ with $\mathcal{P}^* = \{\mathsf{pk}_1 = \mathsf{pk}^*, \ldots, \mathsf{pk}_N\}, \sigma^* = (\mathsf{com}^*, \varphi^*, s^*, B^*)$, and $B^* = b_1^* \ldots b_N^* \in \{0,1\}^N$, it holds that $b_1^* = 1 - b^*$ and $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = b^*$.

If we condition on $\mathbf{G}_1 \Rightarrow 1$, then we claim that b^* and $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$ are uniformly random and independent, and independent of $\mathcal{A}$'s view. In particular, they are independent of b_1^*. This is because bit b^* is hidden from $\mathcal{A}$ by construction, and $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$ is hidden from $\mathcal{A}$ due to $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathsf{Queried}$ and the change introduced in $\mathbf{G}_1$. Therefore, we have

$$\Pr\left[\mathsf{RightBits} \mid \mathbf{G}_1 \Rightarrow 1\right] = \Pr_{b,b^* \xleftarrow{\$} \{0,1\}}\left[b_1^* = 1 - b^* \wedge b = b^*\right] = \frac{1}{4}.$$

With this, we obtain

$$\begin{aligned}\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] &= \Pr\left[\mathsf{RightBits} \wedge \mathbf{G}_1 \Rightarrow 1\right] \\ &= \Pr\left[\mathsf{RightBits} \mid \mathbf{G}_1 \Rightarrow 1\right] \cdot \Pr\left[\mathbf{G}_1 \Rightarrow 1\right] = \frac{1}{4} \cdot \Pr\left[\mathbf{G}_1 \Rightarrow 1\right].\end{aligned}$$

Game $\mathbf{G}_3$: This game is the same as $\mathbf{G}_2$, but we add another abort. Namely, once the game sampled par and $x_{1,0}, x_{1,1}$ at the beginning of the game, it returns 0 and terminates if $(\mathsf{par}, x_{1,1-b^*}) \notin \mathsf{Good}$, where Good is as in the definition of the weakly equivocable coset commitment scheme. Otherwise, it continues as in $\mathbf{G}_2$ does. By the good parameters property of CMT, we have

$$\left|\Pr\left[\mathbf{G}_2 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_3 \Rightarrow 1\right]\right| \leq \Pr\left[(\mathsf{par}, x_{1,1-b^*}) \notin \mathsf{Good}\right] \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_4$: In this game, we change how random oracle H is simulated. Recall that until now, when H is queried on an input $(\langle \mathcal{P} \rangle, \mathsf{m})$ and the output of H is not yet defined, it samples a random commitment key $\mathsf{ck} \xleftarrow{\$} \mathcal{K}$ uniformly at random and defines the output to be this key. From now on, we sample ck differently, distinguishing two cases depending on the bit $b := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$ and the bit b^*. Namely, if $b = 1 - b^*$, then ck is sampled in hiding mode with a trapdoor, i.e., $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1-b^*})$. Further, the trapdoor td is stored in a map tr by setting $tr[\langle \mathcal{P} \rangle, \mathsf{m}] := \mathsf{td}$. On the other hand, if $b = b^*$, then ck is sampled in binding mode, i.e., $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. We now show indistinguishability of $\mathbf{G}_3$ and $\mathbf{G}_4$. First, note that keys sampled in the first case are distributed identically in $\mathbf{G}_3$ and $\mathbf{G}_4$. This follows from the uniform keys property of CMT, which we can apply due to the previous change that ensures that $(\mathsf{par}, x_{1,1-b^*}) \in \mathsf{Good}$. Second, keys sampled in the second case are indistinguishable by the multi-key indistinguishability property of CMT. More precisely, there is a reduction $\mathcal{B}$ that gets as input $\mathsf{par}, x_{1,1-b^*}$, and commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$. It then simulates game $\mathbf{G}_4$ for $\mathcal{A}$, but embedding the commitment keys ck_i whenever random oracle H needs to be simulated and $b = b^*$ as above. In the end, $\mathcal{B}$ outputs whatever the game outputs. Clearly, $\mathcal{B}$'s running time is determined by the running time of $\mathcal{A}$ and it perfectly simulates $\mathbf{G}_4$ if the keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_{Q_{\mathsf{H}}}$ are generated via algorithm BGen. Otherwise, if the keys are sampled uniformly at random, it perfectly simulates $\mathbf{G}_3$ for $\mathcal{A}$. For this, it was important that we introduced the indirection via oracle $\bar{\mathsf{H}}_b$ as otherwise the simulation would not be perfect. Concretely, if $\mathcal{B}$ itself had queried H_b instead of $\bar{\mathsf{H}}_b$, then the game would always have output 0, see the change in $\mathbf{G}_2$. We have

$$\left|\Pr\left[\mathbf{G}_3 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_4 \Rightarrow 1\right]\right| \leq \mathsf{Adv}^{Q_{\mathsf{H}}\text{-}\mathsf{keydist}}_{\mathcal{B},\mathsf{CMT}}(\lambda).$$

Game $\mathbf{G}_5$: In this game, we change the signing oracle. The result will be that we can simulate the signing oracle without $x_{1,1-b^*}$, but using trapdoors for commitment keys instead. First, we explain how we change oracle $\textsc{Sig}_0$, which runs Sig_0 in previous games. Recall from the definition of Sig_0, that this means that the oracle on input $\mathcal{P}$ and m samples $r_1 \xleftarrow{\$} \mathcal{D}$, defines $R_1 := \mathsf{F}(r_1)$, computes a bit $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$ and a commitment key $\mathsf{ck} := \mathsf{H}(\langle \mathcal{P} \rangle, \mathsf{m})$, and commits to R_1 by sampling $\varphi_1 \xleftarrow{\$} \mathcal{G}$ and setting $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$. Now, if $b_1 = b^*$, we don't change anything and $\mathbf{G}_5$ behaves as previous games do. However, if $b_1 = 1 - b^*$, the game computes com_1 differently. It computes it as $(\mathsf{com}_1, St_1) \leftarrow \mathsf{TCom}(\mathsf{ck}, tr[\langle \mathcal{P} \rangle, \mathsf{m}])$. Here, recall that if $b_1 = 1 - b^*$, then ck has been generated with a trapdoor that is stored in tr, see $\mathbf{G}_4$. Next, we explain how we change oracle $\textsc{Sig}_1$, which runs Sig_1 in previous games. To recall, this means that first, a challenge c_1 is computed

using the random oracle H_c and all messages of the first round. Then, a response $s_1 := c_1 \cdot x_{1,b_1} + r_1$ is computed, and s_1 and φ_1 is returned to $\mathcal{A}$. Again, we only change the case where $b_1 = 1 - b^*$. Namely, in this case, the game runs $(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St_1, c_1)$ to compute s_1 and φ_1 instead. Due to the change introduced in $\mathbf{G}_3$, we know that $(\mathsf{par}, x_{1,1-b^*}) \in \mathsf{Good}$, and thus we can apply the weak trapdoor property of CMT for every signing query. We get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq Q_S \varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_6$: In this game, we undo the change from $\mathbf{G}_3$, namely, we no longer require that $(\mathsf{par}, x_{1,1-b^*}) \in \mathsf{Good}$. With a similar argument as in $\mathbf{G}_3$, we get

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \varepsilon_{\mathsf{g}}.$$

Game $\mathbf{G}_7$: In this game, we change how $X_{1,1-b^*}$ is generated. Recall that until now, it is generated by sampling $x_{1,1-b^*} \xleftarrow{\$} \mathcal{D}$ and setting $X_{1,1-b^*} := \mathsf{F}(x_{1,1-b^*})$. From now on, we sample it in lossy mode, i.e., as $X_{1,1-b^*} \xleftarrow{\$} \mathcal{R}$. Observe that $x_{1,1-b^*}$ is used nowhere else during the game, due to our previous changes. Therefore, we can easily bound the distinguishing advantage between $\mathbf{G}_6$ and $\mathbf{G}_7$ by a reduction $\mathcal{B}'$ that runs in the key indistinguishability game of LF and embeds its input in $X_{1,1-b^*}$. We have

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{B}',\mathsf{LF}}(\lambda).$$

Game $\mathbf{G}_8$: In game $\mathbf{G}_8$, we make use of the statistical coset binding property of CMT. Concretely, we change oracle H_c and the winning condition. Recall that until now, a query $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B)$ is answered in the standard way using lazy sampling. In game $\mathbf{G}_8$, this is still the case, but additionally the extractor Ext for the statistical coset binding property of CMT is run in certain cases. Namely, write $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$ and $B = b_1 \ldots b_N$. Further, set $b := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. If pk^* is part of $\mathcal{P}$, i.e., $\mathsf{pk}^* = \mathsf{pk}_1$, and $b = b^*$, then we know that the commitment key $\mathsf{ck} := \mathsf{H}(\langle \mathcal{P} \rangle, \mathsf{m})$ is generated in binding mode by algorithm BGen. This is due to the change in $\mathbf{G}_4$. Now, game $\mathbf{G}_8$ runs $R \leftarrow \mathsf{Ext}(\mathsf{H}(\langle \mathcal{P} \rangle, \mathsf{m}), \mathsf{com})$ and stores R in a map $r[\cdot]$ as $r[\mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B] := R$. Other than that, the oracle H_c does not change. Next, we describe how the winning condition is changed. For that, assume that $\mathcal{A}$ outputs a forgery $\mathcal{A}$ outputs a forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$ with $\mathcal{P}^* = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, $\sigma^* = (\mathsf{com}^*, \varphi^*, s^*, B^*)$, and $B^* = b_1^* \ldots b_N^* \in \{0,1\}^N$. Assume that game $\mathbf{G}_7$ does not return 0. Especially, we have $\mathsf{pk}_1 = \mathsf{pk}^*$ and $(\mathcal{P}^*, \mathsf{m}^*) \notin \mathsf{Queried}$, and $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = b^*$ (see $\mathbf{G}_2$). Further, the game parses $\mathsf{pk}_i = (X_{i,0}, X_{i,1})$ for every key pk_i in $\mathcal{P}^*$ and defines challenges $c_i^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*)$ for all $i \in [N]$ as the verification algorithm does. In particular, now we know, due to $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = b^*$, that $r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$ is defined. Next, the game defines $R^* := \mathsf{F}(s^*) - \sum_{i=1}^N c_i^* \cdot X_{i,b_i^*}$ as the verification algorithm does. The game outputs 0 if $R^* \notin r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*] + \mathsf{F}(\mathcal{D})$. Otherwise, it behaves as $\mathbf{G}_7$ does. Note that if $\mathbf{G}_7$ outputs 1, but $\mathbf{G}_8$ does not, then we know that $\mathsf{com}^* = \mathsf{Com}(\mathsf{ck}, R^*; \varphi^*)$, where $\mathsf{ck} := \mathsf{H}(\langle \mathcal{P} \rangle^*, \mathsf{m}^*)$. In other words, $\mathbf{G}_7$ and $\mathbf{G}_8$ only differ, if for the forgery, the value $r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$ that the extractor extracted from commitment com^* is in a different coset than the value to which $\mathcal{A}$ successfully opens com^* in its forgery. We can easily bound the probability of this using the statistical coset binding property of CMT. For that, we sketch an (unbounded) reduction that gets as input the parameters par of the linear function, and a commitment key $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$. Then, it first samples indices $i_{\mathsf{H}} \xleftarrow{\$} [Q_{\mathsf{H}}]$ and $i_{\mathsf{H}_c} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$ uniformly at random, and then simulates the game $\mathbf{G}_8$ honestly for $\mathcal{A}$, except the i_{H}th query to H and the i_{H_c}th query to H_c. In the i_{H}th query to H, if it has to sample a binding key, it embeds ck. Otherwise, it aborts. In the i_{H_c}th query to H_c, if it had to run Ext, it instead outputs the commitment com and its state to the statistical coset binding game. Otherwise, it aborts. Finally, when $\mathcal{A}$ outputs its forgery, and the i_{H}th query to H and the i_{H_c}th query to H_c are the queries of interest, and $R^* \notin r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*] + \mathsf{F}(\mathcal{D})$, the reduction outputs R^* and φ^*, thereby winning the statistical coset binding game. It is easy to see that this shows

$$|\Pr[\mathbf{G}_7 \Rightarrow 1] - \Pr[\mathbf{G}_8 \Rightarrow 1]| \leq Q_{\mathsf{H}} Q_{\mathsf{H}_c} \varepsilon_{\mathsf{b}}.$$

To finish the proof, we bound the probability that $\mathbf{G}_8$ outputs 1 using coset lossy soundness of LF. For that consider the following unbounded reduction:

- The reduction gets as input parameters par and an element $X \in \mathcal{R}$.
- It picks a random index $\hat{i} \xleftarrow{\$} [Q_{\mathsf{H}_c}]$. It then simulates game $\mathbf{G}_8$ for $\mathcal{A}$ until $\mathcal{A}$ outputs its forgery, using parameters par and defining $X_{1,1-b^*} := X$ instead of picking it randomly from $\mathcal{R}$. Further, the reduction handles the $\hat{i}$th query to H_c differently.
- Let $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B)$ be the $\hat{i}$th query to H_c. If the hash value for this query is already defined, the reduction continues as $\mathbf{G}_8$ would do. Otherwise, let $\mathcal{P} = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\}$, $B = b_1 \ldots b_N$, and $b := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$. The reduction also continues as $\mathbf{G}_8$ would do if pk^* is not in $\mathcal{P}$. Otherwise, assume that $\mathsf{pk}^* = \mathsf{pk}_1$ as usual. If $\mathsf{pk} \neq \mathsf{pk}^*$ or $b \neq b^*$, the reduction also continues as $\mathbf{G}_8$ would do. In other words, the reduction only differs in the case in which $\mathbf{G}_8$ would run the extractor Ext. In this case, the reduction runs Ext as $\mathbf{G}_8$ would do, i.e., it runs $\hat{R} \leftarrow \mathsf{Ext}(\mathsf{H}(\langle \mathcal{P} \rangle, \mathsf{m}), \mathsf{com})$ and sets $r[\mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B] := \hat{R}$. In addition, the reduction sets $c_i := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B)$ for each $i \in [N] \setminus \{1\}$, and computes

  $$R := \hat{R} + \sum_{i=2}^{N} c_i \cdot X_{i,b_i}.$$

  Then, the reduction outputs R to the coset lossy soundness game, and in return it receives a challenge $c \in \mathcal{S}$. Finally, the reduction programs $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B] := c$ and returns this hash value.
- When $\mathcal{A}$ outputs its forgery $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$, the reduction does all the verification checks as in $\mathbf{G}_8$. Assuming all of these checks pass, write $\mathcal{P}^* = \{\mathsf{pk}_1 = \mathsf{pk}^*, \ldots, \mathsf{pk}_N\}$, $\sigma^* = (\mathsf{com}^*, \varphi^*, s^*, B^*)$, and $B^* = b_1^* \ldots b_N^* \in \{0,1\}^N$. The reduction aborts if the value $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*)$ has not been defined during the $\hat{i}$th query to H_c. Otherwise, the reduction returns $s := s^*$ to the coset lossy soundness game.

One can easily see that the view of $\mathcal{A}$ is independent of the index $\hat{i}$ until a potential abort, and that, assuming the reduction does not abort, the simulation of $\mathbf{G}_8$ is perfect. Now, we want to argue that the reduction breaks coset lossy soundness if $\mathbf{G}_8$ outputs 1, and the index $\hat{i}$ is guessed correctly. Once this is shown, we can conclude with

$$\Pr[\mathbf{G}_8 \Rightarrow 1] \leq Q_{\mathsf{H}_c} \varepsilon_{\mathsf{l}}.$$

To show this claim, we assume that $\mathbf{G}_8$ outputs 1 and the index $\hat{i}$ is guessed correctly. Now, it follows from the condition $\bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = b^*$ introduced in $\mathbf{G}_2$ that the reduction output R as above to the coset lossy soundness game, received c, and programmed $\mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*)$ to be c. It remains to argue that $\mathsf{F}(s) - c \cdot X \in R + \mathsf{F}(\mathcal{D})$. For that, first recall that the change introduced in $\mathbf{G}_8$ ensures that

$$\mathsf{F}(s^*) - \sum_{i=1}^{N} c_i^* \cdot X_{i,b_i^*} \in r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*] + \mathsf{F}(\mathcal{D}).$$

Using the assumption that the index $\hat{i}$ is guessed correctly, this implies

$$\mathsf{F}(s^*) - c \cdot X_{1,b_1^*} - \sum_{i=2}^{N} c_i \cdot X_{i,b_i^*} \in \hat{R} + \mathsf{F}(\mathcal{D}).$$

Now, we rearrange terms and use the condition $b_1^* = 1 - b^*$ introduced in $\mathbf{G}_2$, and get

$$\mathsf{F}(s^*) - c \cdot X_{1,1-b^*} \in \hat{R} + \sum_{i=2}^{N} c_i \cdot X_{i,b_i^*} + \mathsf{F}(\mathcal{D}).$$

If we recall the definition of $s^* = s$, $X_{1,1-b^*} = X$, and the definition of R, then this is exactly the statement we want to show. Concluded. □

4.5.4 Instantiation

We have presented abstract constructions of two-round multi-signatures, namely, $\mathsf{ToothKA}[\mathsf{LF}, \mathsf{CMT}]$ in Section 4.5.2 and $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}]$ in Section 4.5.3. To obtain concrete constructions Toothpicks I and Toothpicks II, we instantiate the linear function family LF and the commitment scheme CMT both based on DDH. Our instantiation can be interpreted as an optimization of the one in Section 4.4.4. We exploit the fact that the commitment scheme only has to be binding with respect to cosets.

Linear Function Family. We use the same linear function family $\mathsf{LF}_{\mathsf{DDH}}$ as in Section 4.4.4. To recall, $\mathsf{LF}_{\mathsf{DDH}} = (\mathsf{Gen}, \mathsf{F})$ is based on the DDH assumption. Algorithm Gen runs GGen to obtain the description of a prime order group $\mathbb{G}$ of order p with generator g. It also samples an element $h \xleftarrow{\$} \mathbb{G}$ and outputs parameters $\mathsf{par} := (g, h) \in \mathbb{G}^2$. The description of $\mathbb{G}$ is also contained in par and left implicit for the sake of a concise presentation. These parameters define the set of scalars, domain, range, and the function $\mathsf{F}(\mathsf{par}, \cdot)$, which are as follows:

$$\mathcal{S} := \mathbb{Z}_p, \;\; \mathcal{D} := \mathbb{Z}_p, \;\; \mathcal{R} := \mathbb{G} \times \mathbb{G}, \;\; \mathsf{F}(\mathsf{par}, x) := (g^x, h^x).$$

One can easily verify that this is a linear function family. Further, we have shown in Section 4.4.4 that $\mathsf{LF}_{\mathsf{DDH}}$ satisfies key indistinguishability, lossy soundness, and aggregation lossy soundness. Using Lemmas 4.10 and 4.11, we conclude that $\mathsf{LF}_{\mathsf{DDH}}$ satisfies coset lossy soundness and coset aggregation lossy soundness. The following lemma summarizes this.

Lemma 4.14. *Assuming that the* DDH *assumption holds relative to* GGen*, the linear function family* $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies key indistinguishability. Concretely, for any PPT algorithm* $\mathcal{A}$ *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{\mathsf{keydist}}_{\mathcal{A},\mathsf{LF}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda).$$

Further, the linear function family $\mathsf{LF}_{\mathsf{DDH}}$ *satisfies* ε_{l}*-coset lossy soundness and* $\varepsilon_{\mathsf{al}}$*-coset aggregation lossy soundness for*

$$\varepsilon_{\mathsf{l}} \leq 3/p, \;\; \varepsilon_{\mathsf{al}} \leq 4/p.$$

Commitment Scheme. Next, we present our instantiation of the weakly equivocable coset commitment scheme for the linear function family $\mathsf{LF}_{\mathsf{DDH}}$ introduced before. Our commitment scheme shares similarities with the commitment scheme we have constructed in Section 4.4.4, which uses a 3×3 matrix of group elements as a commitment key. Our crucial observation is that if we replace this 3×3 structure with a more efficient 2×2 structure, we obtain a scheme that is still binding on cosets. We now describe our commitment scheme $\mathsf{CMT}_{\mathsf{DDH}} = (\mathsf{BGen}, \mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$ for $\mathsf{LF}_{\mathsf{DDH}}$. Assume parameters of $\mathsf{LF}_{\mathsf{DDH}}$ are given, specifying a group $\mathbb{G}$. Then, the commitment scheme has key space $\mathcal{K} := \mathbb{G}^{2\times 2}$, message space $\mathcal{D} = \mathbb{G} \times \mathbb{G}$, randomness space $\mathcal{G} = \mathbb{Z}_p^2$, and commitment space $\mathcal{H} = \mathbb{G}^2$. The spaces $\mathcal{D}, \mathcal{G}$, and $\mathcal{H}$ are associated with the natural componentwise group operations. Next, we describe the algorithms of the commitment scheme verbally.

- $\mathsf{BGen}(\mathsf{par}) \to \mathsf{ck}$: Parse $\mathsf{par} = (g, h)$. Sample $a, b \xleftarrow{\$} \mathbb{Z}_p$ and set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} := \begin{pmatrix} g^a & g^b \\ h^a & h^b \end{pmatrix} \in \mathbb{G}^{2\times 2}.$$

- $\mathsf{TGen}(\mathsf{par}, X = (X_1, X_2)) \to (\mathsf{ck}, \mathsf{td})$: Sample exponents $d_{i,j} \xleftarrow{\$} \mathbb{Z}_p$ for all $(i, j) \in [2] \times [2]$. Set

$$\mathsf{ck} := \mathbf{A} := \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} := \begin{pmatrix} X_1^{d_{1,1}} & X_1^{d_{1,2}} \\ X_2^{d_{2,1}} & X_2^{d_{2,2}} \end{pmatrix} \in \mathbb{G}^{2\times 2}.$$

  Further, set $\mathsf{td} := (\mathbf{D}, X_1, X_2)$ for

$$\mathbf{D} := \begin{pmatrix} d_{1,1} & d_{1,2} \\ d_{2,1} & d_{2,2} \end{pmatrix} \in \mathbb{Z}_p^{2\times 2}.$$

- $\mathsf{Com}(\mathsf{ck}, R = (R_1, R_2); \varphi) \to \mathsf{com}$: Let $\varphi = (\alpha, \beta) \in \mathbb{Z}_p^2$. Compute $\mathsf{com} := (C_1, C_2)$ for

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} R_1 \cdot & A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta} \\ R_2 \cdot & A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta} \end{pmatrix}.$$

- $\mathsf{TCom}(\mathsf{ck}, \mathsf{td}) \to (\mathsf{com}, St)$: Sample $\rho_1, \rho_2, s \xleftarrow{\$} \mathbb{Z}_p$. Set $St := (\mathsf{td}, \tau, \rho_1, \rho_2, s)$ and compute $\mathsf{com} := (C_1, C_2)$ for

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} := \begin{pmatrix} X_1^{\rho_1} \cdot g^s \\ X_2^{\rho_2} \cdot h^s \end{pmatrix}.$$

- $\mathsf{TCol}(St, c) \to (\varphi, R, s)$: Set $R := (R_1, R_2) := \left(g^s \cdot X_1^{-c}, h^s \cdot X_2^{-c}\right)$. Then, if $\mathbf{D}$ is not invertible, return $\perp$. Otherwise, compute $\varphi := (\alpha, \beta)$ for

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \mathbf{D}^{-1} \cdot \begin{pmatrix} \rho_1 + c \\ \rho_2 + c \end{pmatrix}.$$

Theorem 4.6. *If the* DDH *assumption holds relative to* GGen*, then* $\mathsf{CMT}_{\mathsf{DDH}}$ *is an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, with*

$$\varepsilon_{\mathsf{b}} = 0, \quad \varepsilon_{\mathsf{g}} \leq 2/p, \quad \varepsilon_{\mathsf{t}} \leq 2/p.$$

Concretely, for any PPT algorithm $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{2Q\text{-DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda)..$$

The proof of Theorem 4.6 is split into a sequence of lemmas, showing the required properties of a weakly equivocable coset commitment scheme separately. The homomorphism property is easy to verify. For the remaining properties, we first have to define a set Good. We define it as the set of non-zero parameters and domain elements, i.e.,

$$\mathsf{Good} = \left\{((g,h),x) \in \mathbb{G}^2 \times \mathbb{Z}_p \;\middle|\; (g,h) \in \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda) \wedge h \neq g^0 \wedge x \neq 0\right\}.$$

It is clear that membership in Good can be decided efficiently. Further, for $(g,h) \leftarrow \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda)$ and $x \xleftarrow{\$} \mathbb{Z}_p$, the probability that $((g,h),x) \notin \mathsf{Good}$ is at most $2/p$ by a union bound over the two events $h = g^0$ and $x = 0$. This shows $\varepsilon_{\mathsf{g}} \leq 2/p$. We proceed by showing that the commitment scheme satisfies the uniform keys property, the weak trapdoor property, multi-key indistinguishability, and is statistically coset binding. The proofs of the uniform keys property and the weak trapdoor property are similar to the proofs of the corresponding statement in Section 4.4.4.

Lemma 4.15. *The scheme* $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the uniform keys property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*.*

Proof. Let $((g,h),x) \in \mathsf{Good}$ and define $(X_1, X_2) \in \mathbb{G}^2$ to be the image of x under F, i.e., $X_1 = g^x$ and $X_2 = h^x$. We have to argue that the distribution of $((g,h),x,\mathbf{A})$ for a commitment key $\mathbf{A}$ as output by algorithm TGen is the same as for a random $\mathbf{A} \xleftarrow{\$} \mathbb{G}^{2\times 2}$. Recall that $\mathbf{A}$ output by TGen has the form

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} = \begin{pmatrix} X_1^{d_{1,1}} & X_1^{d_{1,2}} \\ X_2^{d_{2,1}} & X_2^{d_{2,2}} \end{pmatrix} \in \mathbb{G}^{2\times 2},$$

where the $d_{i,j}$ are sampled uniformly at random from $\mathbb{Z}_p$. As $((g,h),x) \in \mathsf{Good}$, we know that X_1 and X_2 are generators of $\mathbb{G}$, and therefore $\mathbf{A}$ is uniformly random over $\mathbb{G}^{2\times 2}$. □

Lemma 4.16. *The scheme* $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the weak trapdoor property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, where* $\varepsilon_{\mathsf{t}} \leq 2/p$*.*

Proof. Fix parameters $g, h \in \mathbb{G}$ and $x \in \mathbb{Z}_p$ and a challenge $c \in \mathbb{Z}_p$ such that $((g,h),x) \in \mathsf{Good}$. Define $(X_1, X_2) \in \mathbb{G}^2$ to be the image of x under F, i.e., $X_1 = g^x$ and $X_2 = h^x$. According to the definition of the weak trapdoor property, we have to consider two different distributions $\mathcal{T}_0$ and $\mathcal{T}_1$ of tuples

$$((g,h), \mathbf{A}, (\mathbf{D}, X_1, X_2), x, c, (C_1, C_2), ((\alpha,\beta),(R_1,R_2),s))$$

Here, g, h, x, c, X_1, X_2 are fixed as above and $\mathbf{A}, \mathbf{D}, X_1, X_2$ are output by TGen in both distributions $\mathcal{T}_0$ and $\mathcal{T}_1$. In distribution $\mathcal{T}_1$, the remaining components $(C_1, C_2), ((\alpha,\beta),(R_1,R_2),s$ are sampled via

$$((C_1,C_2),St) \leftarrow \mathsf{TCom}(\mathsf{ck},\mathsf{td}),\ ((\alpha,\beta),(R_1,R_2),s) \leftarrow \mathsf{TCol}(St,c).$$

In distribution $\mathcal{T}_0$, the remaining components $(C_1, C_2), ((\alpha,\beta),(R_1,R_2),s$ are sampled as

$$\begin{gathered} r \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^r,\ R_2 := h^r,\ s := c \cdot x + r, \\ \alpha,\beta \xleftarrow{\$} \mathbb{Z}_p,\ (C_1,C_2) := \mathsf{Com}(\mathbf{A},(R_1,R_2);(\alpha,\beta)). \end{gathered}$$

We will now gradually change this distribution $\mathcal{T}_0$ until we arrive at the distribution $\mathcal{T}_1$. Before we do that, we assume that in both distributions $\mathcal{T}_0$ and $\mathcal{T}_1$, the matrix $\mathbf{D} \in \mathbb{Z}_p^{2\times 2}$ has full rank. As $\mathbf{D}$ is sampled uniformly at random over $\mathbb{Z}_p^{2\times 2}$, we know that it has full rank except with probability $1/p$. Thus, this assumption will add $2/p$ to our final bound.

In our first step, we remove the dependence on r and x. That is, we define a new distribution, in which the remaining components $(C_1, C_2), ((\alpha,\beta),(R_1,R_2),s$ are samples as

$$\begin{gathered} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \alpha,\beta \xleftarrow{\$} \mathbb{Z}_p,\ (C_1,C_2) := \mathsf{Com}(\mathbf{A},(R_1,R_2);(\alpha,\beta)). \end{gathered}$$

It is clear that $\mathcal{T}_0$ and this new distribution are identical[10]. Next, we want to make (C_1, C_2) independent of (R_1, R_2). For that, we consider the mapping from (α,β) to (C_1, C_2) induced by Com, namely,

$$\Psi\colon \mathbb{Z}_p^2 \to \mathbb{G}^2,\ \ (\alpha,\beta) \mapsto (C_1,C_2) = (R_1 \cdot A_{1,1}^{\alpha} \cdot A_{1,2}^{\beta}, R_2 \cdot A_{2,1}^{\alpha} \cdot A_{2,2}^{\beta}).$$

Using the assumption that $\mathbf{D}$ has full rank, that g, h, X_1, X_2 are generators of $\mathbb{G}$, and the definition of the $A_{i,j}$, we see that Ψ is a bijection, and (C_1, C_2) is uniformly random over $\mathbb{G}^2$. Therefore, we can equivalently write the distribution as

$$\begin{gathered} s \xleftarrow{\$} \mathbb{Z}_p,\ R_1 := g^s \cdot X_1^{-c},\ R_2 := h^s \cdot X_2^{-c}, \\ \rho_1,\rho_2 \xleftarrow{\$} \mathbb{Z}_p,\ (C_1,C_2) := (X_1^{\rho_1} \cdot g^s, X_2^{\rho_2} \cdot h^s),\ \ (\alpha,\beta) := \Psi^{-1}(C_1,C_2). \end{gathered}$$

We claim that this is exactly the distribution $\mathcal{T}_1$. For that, it is sufficient to argue that the way algorithm TCol computes (α,β), i.e., as $(\alpha,\beta)^t := \mathbf{D}^{-1}(\rho_1 + c, \rho_2 + c)^t$, is identical to Ψ^{-1}. By definition of Ψ, the expression $(\alpha,\beta) = \Psi^{-1}(C_1,C_2)$ is equivalent to

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} R_1 \cdot & A_{1,1}^{\alpha} \cdot & A_{1,2}^{\beta} \\ R_2 \cdot & A_{2,1}^{\alpha} \cdot & A_{2,2}^{\beta} \end{pmatrix}.$$

Using our definition of $\mathbf{A}$, C_1, C_2, and R_1, R_2, this is equivalent to

$$\begin{pmatrix} X_1^{\rho_1} \cdot g^s \\ X_2^{\rho_2} \cdot h^s \end{pmatrix} = \begin{pmatrix} g^s \cdot X_1^{-c} \cdot & X_1^{d_{1,1}\alpha} \cdot & X_1^{d_{1,2}\beta} \\ h^s \cdot X_2^{-c} \cdot & X_2^{d_{2,1}\alpha} \cdot & X_2^{d_{2,2}\beta} \end{pmatrix}.$$

[10] Essentially, we used the special honest-verifier zero-knowledge property of the Chaum-Pedersen identification scheme [CP93, KW03, KMP16] here.

The g^s and h^s terms cancel out, and this is equivalent to

$$\begin{pmatrix}\rho_1 + c\\ \rho_2 + c\end{pmatrix} = \mathbf{D} \cdot \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$$

which concludes the proof. □

Lemma 4.17. *The scheme* $\mathsf{CMT}_{\mathsf{DDH}}$ *satisfies the statistical coset binding property of an* $(\varepsilon_{\mathsf{b}}, \varepsilon_{\mathsf{g}}, \varepsilon_{\mathsf{t}})$*-weakly equivocable coset commitment scheme for* $\mathsf{LF}_{\mathsf{DDH}}$*, where* $\varepsilon_{\mathsf{b}} = 0$*.*

Proof. To show that the scheme is statistically coset binding, we have to present an algorithm Ext that outputs (R_1, R_2) on input $\mathbf{A} \in \mathbb{Z}_p^{2\times 2}$ and $\mathsf{com} = (C_1, C_2) \in \mathbb{G}^2$. Algorithm Ext just outputs $(R_1, R_2) := (C_1, C_2)$. It remainst to show that Ext satisfies the conditions of the statistical coset binding property. Concretely, we have to consider the following experiment for any adversary $\mathcal{A}$: First, parameters $\mathsf{par} := (g, h) \leftarrow \mathsf{LF}_{\mathsf{DDH}}.\mathsf{Gen}(1^\lambda)$ and a commitment key $\mathsf{ck} := \mathbf{A} \leftarrow \mathsf{BGen}(\mathsf{par})$ are generated. Then, $\mathcal{A}$ gets par and ck and outputs a commitment $\mathsf{com} = (C_1, C_2) \in \mathbb{G}^2$. The extractor Ext is run, outputting $R_1 = C_1, R_2 = C_2$. Finally, $\mathcal{A}$ outputs $(R_1', R_2') \in \mathbb{G}^2$ and $(\alpha', \beta') \in \mathbb{Z}_p^2$. We have to bound the probability of the event

$$\mathsf{Com}(\mathsf{ck}, (R_1', R_2'); (\alpha', \beta')) = \mathsf{com} \ \wedge \ \forall \delta \in \mathbb{Z}_p : \begin{pmatrix}R_1'\\ R_2'\end{pmatrix} \neq \begin{pmatrix}R_1 \cdot g^\delta\\ R_2 \cdot h^\delta\end{pmatrix}.$$

Now, we rewrite the first condition according to the definition of Com, taking into account the definition of $\mathbf{A}$ in algorithm BGen. Further, we use $(R_1, R_2) = (C_1, C_2)$. Then, this is equivalent to

$$\begin{pmatrix}R_1' \cdot & g^{a\cdot\alpha'} \cdot g^{b\cdot\beta'}\\ R_2' \cdot & h^{a\cdot\alpha'} \cdot h^{b\cdot\beta'}\end{pmatrix} = \begin{pmatrix}R_1\\ R_2\end{pmatrix} \ \wedge \ \forall \delta \in \mathbb{Z}_p : \begin{pmatrix}R_1'\\ R_2'\end{pmatrix} \neq \begin{pmatrix}R_1 \cdot g^\delta\\ R_2 \cdot h^\delta\end{pmatrix}.$$

Now, by taking $\delta = -(a \cdot \alpha' + b \cdot \beta')$, it is easy to see that this can never hold, and therefore the event we have to bound can never occur. □

Lemma 4.18. *For any PPT algorithm* $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{Q\text{-}\mathsf{keydist}}_{\mathcal{A},\mathsf{CMT}_{\mathsf{DDH}}}(\lambda) \leq \mathsf{Adv}^{2Q\text{-}\mathsf{DDH}}_{\mathcal{A},\mathsf{GGen}}(\lambda).$$

Proof. To prove multi-key indistinguishability based on the $2Q$-DDH assumption, we assume the existence of an adversary $\mathcal{A}$ against the multi-key indistinguishability of $\mathsf{CMT}_{\mathsf{DDH}}$, and turn it into an algorithm that breaks the $2Q$-DDH assumption. That is, we construct a reduction $\mathcal{B}$ that simulates the multi-key indistinguishability game for $\mathcal{A}$ and runs in the $2Q$-DDH game. Reduction $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ gets as input $\mathbb{G}, p, g, h$ and $2Q$ group elements $(u_i, v_i)_{i=1}^{2Q}$.
2. $\mathcal{B}$ defines $\mathsf{par} := (g, h)$ and samples $x \xleftarrow{\$} \mathbb{Z}_p$. If $((g, h), x) \notin \mathsf{Good}$, $\mathcal{B}$ returns 0 and terminates.
3. Otherwise, $\mathcal{B}$ defines Q commitment keys $\mathsf{ck}_1, \ldots, \mathsf{ck}_Q \in \mathbb{G}^{2\times 2}$ via

$$\mathsf{ck}_i := \begin{pmatrix}u_{2i-1} & u_{2i}\\ v_{2i-1} & v_{2i-1}\end{pmatrix} \text{ for all } i \in [Q].$$

4. $\mathcal{B}$ runs $\mathcal{A}$ on input par, x and $(\mathsf{ck}_i)_{i\in[Q]}$. It returns whatever $\mathcal{A}$ returns.

It is clear that the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. Further, assume that $\mathcal{B}$'s input satisfies $u_i = g^{a_i}, v_i = g^{a_i}$ for random $a_i \in \mathbb{Z}_p$. In this case, the commitment keys are distributed as if they are generated by BGen, and $\mathcal{B}$ perfectly simulates game $Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{0,\mathsf{CMT}}(\lambda)$ for $\mathcal{A}$. On the other hand, if $u_i, v_i \xleftarrow{\$} \mathbb{G}$ for all $i \in [2Q]$, then the resulting commitment keys ck_i are distributed uniformly over $\mathbb{G}^{2\times 2}$, and $\mathcal{B}$ perfectly simulates game $Q\text{-}\mathbf{KEYDIST}^{\mathcal{A}}_{1,\mathsf{CMT}}(\lambda)$ for $\mathcal{A}$. This shows the claim. □

4.6 Concrete Parameters and Efficiency

We have presented four constructions of multi-signatures in this chapter, namely, Chopsticks I and Chopsticks II in Section 4.4 and Toothpicks I and Toothpicks II in Section 4.5. Here, we discuss the efficiency of these schemes. We first explain minor optimizations that improve signature size and communication complexity, and then discuss the asymptotic and concrete efficiency of our schemes.

Further Optimizations. We describe the optimizations for our tight construction in Section 4.5.3, i.e., for Toothpicks II, but they directly translate to our other constructions as well. Our first optimization is to reduce the communication complexity by deriving the commitment randomness φ, which consists of two (resp. three) field elements, from a short seed of length λ bit using a random oracle $\bar{\mathsf{H}}\colon \{0,1\}^* \to \mathbb{Z}_p^2$. Then, instead of sending φ in the second round, each signer sends its seed, and the signers locally derive all φ's and aggregate them. By the unpredictability of the random oracle, the scheme stays secure. Our second optimization allows us to remove the commitment (resp. commitments) com, which consists of two (resp. three) group elements, from the final signature. The idea is to replace it by a hash of com of length 2λ using another random oracle $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$. Concretely, the signers first complete the signing protocol as before, but to define signer specific challenges, they compute $c_i := \mathsf{H}_c(\mathsf{pk}_i, h, \mathsf{m}, \langle \mathcal{P} \rangle, B)$ for all $i \in [N]$, where $h := \hat{\mathsf{H}}(\mathsf{com})$. In the end, the signature is $\sigma := (h, \varphi, s, B)$. The signature is verified by first recomputing the c_i's as before (using h instead of com) and by checking the commitment after hashing, i.e., using the equation

$$h = \hat{\mathsf{H}}\left(\mathsf{Com}\left(\mathsf{ck}, \mathsf{F}(s) - \sum_{i=1}^{N} c_i \cdot X_{i,b_i}; \varphi\right)\right).$$

In a security proof, we would use the collision-resistance and observability of $\hat{\mathsf{H}}$ to reduce this equation to the original verification equation.

Asymptotics. In Table 4.2, we compare the asymptotic sizes of public keys and signatures and the communication complexity per signer for our schemes with previous schemes in the pairing-free setting. We see that, as expected, Toothpicks I and Toothpicks II outperform Chopsticks I and Chopsticks II in terms of efficiency. Additionally, their communication complexity is smaller than for some non-tight competitors such as Musig2, while the signature size is minimally larger. In general, the asymptotic efficiency of the Toothpicks schemes is comparable with most non-tight or three-round schemes.

Concrete Parameters. We compare the concrete efficiency and security level of two-round multi-signatures in the pairing-free setting in Table 4.3. The numbers are computed using a Python script[11]. Our comparison assumes that all constructions are instantiated with the secp256k1 curve, and we assume security parameter $\lambda = 128$. We compute the concrete security level based on the security bounds (see Table 4.1) assuming the underlying assumption is 128 bit hard, and assuming $Q_H = 2^{30}$ hash queries and $Q_S = 2^{20}$ signing queries. The concrete results confirm what we have seen in the asymptotic comparison.

[11] The Python script can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.

5

Threshold Signatures

Publication History of This Chapter

This chapter is based on the publication [BLT+24] and its full version [BLT+23]. I am the main author of it. To obtain a consistent structure, minor changes to the structure of the paper and minor notational changes have been made.

5.1 Introduction

A threshold signature scheme [Des88, DF90, Ped91] enables a group of n signers to jointly sign a message as long as more than t of them participate. To this end, each of the n signers holds a share of the secret key associated with the public key of the group. When $t+1$ of them come together and run a signing protocol for a particular message, they obtain a compact signature (independent in size of t and n) without revealing their secret key shares to each other. On the other hand, no subset of at most t potentially malicious signers can generate a valid signature. Despite being a well-studied cryptographic primitive, threshold signatures have experienced a renaissance due to their use in cryptocurrencies [LN18] and other modern applications [DOK+20]. This new attention has also led to ongoing standardization efforts [BP22]. In this chapter, we study threshold signatures in the pairing-free discrete logarithm setting. As noted in previous works [TZ22, TZ23, CKM+23b], pairings are not supported in popular libraries and are substantially more expensive to compute, which makes pairing-free solutions appealing.

Static vs. Adaptive Security. When defining security for threshold signatures, the adversary is allowed to concurrently interact with honest signers in the signing protocol. Additionally, it may corrupt up to t out of n parties, thereby learning their secret key material and internal state. Here, we distinguish between *static* corruptions and *adaptive* corruptions. For static corruptions, the adversary declares the set of corrupted parties ahead of time before any messages have been signed. For adaptive corruptions, the adversary can corrupt parties dynamically, depending on previous signatures and corruptions. Adaptive security is a far stronger notion than static security and matches reality more closely. Unfortunately, proving adaptive security for threshold signatures is highly challenging and previous works in the pairing-free setting rely on strong interactive assumptions to simulate the state of adaptively corrupted parties [CKM23a]. This simulation strategy, however, is at odds with rewinding the adversary as part of a security proof. Roughly, if the adversary is allowed to corrupt up to t_c parties, then in the two runs induced by rewinding, it may corrupt up to $2t_c$ parties in total. Thus, for the reduction to obtain meaningful information from the adversary's forgery, it has to be restricted to corrupt at most $t_c \leq t/2$ parties [CKM23a]. To bypass this unnatural restriction, prior work heavily relies on the algebraic group model (AGM) [FKL18] in order to avoid rewinding[1]. In summary: to support an arbitrary corruption threshold, one has to use the AGM or sacrifice adaptive security.

Our Goal. Motivated by this unsatisfactory state of affairs, we want to construct a threshold signature scheme in the pairing-free discrete logarithm setting supporting up to t adaptive corruptions without using the AGM.

5.1.1 Contribution: Twinkle

We construct Twinkle. It is the first threshold signature scheme in the pairing-free setting which combines all of the following characteristics:

- *Adaptive Security.* We prove Twinkle secure under adaptive corruptions. Notably, we do not rely on secure erasures of private state.
- *Non-Interactive Assumptions.* Our security proof relies on a non-interactive and well-studied assumption, namely, the DDH assumption. As a slightly more efficient alternative, we give an instantiation based on a one-more variant of CDH.
- *No AGM.* We do not use the algebraic group model, but only the random oracle model.
- *Arbitrary Threshold.* Twinkle supports an arbitrary corruption threshold $t < n$ for n parties. Essentially, this is established by giving a proof without rewinding.

[1] Other works resort to heavier machinery such as broadcast channels or non-committing encryption resulting in inefficient protocols.

Scheme	Rounds	Adaptive	Assumption	Idealization	Corruptions
GJKR [GJKR07]/StiStr [SS01]	≥ 4	✗	DLOG	ROM	$\leq t < n/2$
Lin-UC [Lin22]	3	✗	DLOG	ROM	$\leq t$
Frost [KG20]	2	✗	DLOG	Custom	$\leq t$
Frost [KG20, BTZ22, BCK+22]	2	✗	AOMDL	ROM	$\leq t$
Frost2 [CKM21, BTZ22, BCK+22]	2	✗	AOMDL	ROM	$\leq t$
Frost3 [RRJ+22]/Olaf [CGRS23]	2	✗	AOMDL	ROM	$\leq t$
TZ [TZ23]	2	✗	DLOG	ROM	$\leq t$
Sparkle [CKM23a]	3	✗	DLOG	ROM	$\leq t$
Sparkle [CKM23a]	3	✓	AOMDL	ROM	$\leq t/2$
Sparkle [CKM23a]	3	✓	AOMDL	ROM+AGM	$\leq t$
Twinkle (AOMCDH)	3	✓	AOMCDH	ROM	$\leq t$
Twinkle (DDH)	3	✓	DDH	ROM	$\leq t$

Table 5.1: Comparison of different threshold signature schemes in the discrete logarithm setting without pairings and the two instantiations of our Twinkle scheme. We compare whether the schemes are proven secure under adaptive corruptions and under which assumption and idealized model they are proven. We also compare the corruption thresholds that they support. For all schemes, we assume that there is a trusted dealer distributing key shares securely. For GJKR [GJKR07]/StiStr [SS01], broadcast channels are assumed, which adds rounds when implemented.

For a comparison of schemes in the pairing-free discrete logarithm setting, see Table 5.1. We also emphasize that we achieve our goal without the use of heavy cryptographic techniques, and our scheme is practical. For example, signatures of Twinkle (from DDH) are at most 3 times as large as regular Schnorr signatures [Sch91], and Twinkle has three rounds. In the context of our proof, we also identify a gap in the analysis of Sparkle [CKM23a] and develop new proof techniques to fix it in the context of our scheme[2].

Conceptually, the design of our threshold signature is inspired by five-move identification schemes, which already have found use in the construction of tightly secure signature schemes [Che05, GJKW07, KLP17]. We achieve our result in two main steps:

1. We first phrase our scheme abstractly using (a variant of) linear function families [HKL19, KLR21, CAHL+22a, PW23a, TZ23]. To prove security under adaptive corruptions, we define a security notion for linear functions resembling a one-more style CDH assumption. This is the step where we identify the gap in the analysis of Sparkle [CKM23a].
2. We then instantiate the linear function family such that this one-more notion follows from the (non-interactive) DDH assumption. Note that Tessaro and Zhu [TZ23] showed a related statement, namely, that a suitable one-more variant of DLOG follows from DLOG. In this sense, our work makes a further step in an agenda aimed at replacing interactive assumptions with non-interactive ones. We are confident that this is interesting in its own right.

5.1.2 More on Related Work

We discuss further related work, including threshold signatures from other assumptions and related cryptographic primitives.

Techniques for Adaptive Security. General techniques for achieving adaptive security have been studied [CGJ+99, JL00, LP01]. Unfortunately, these techniques often rely on heavy cryptographic machinery and assumptions, e.g., secure erasures or broadcast channels.

[2]We communicated the gap and our solution to the authors of Sparkle. To be clear, we do not claim that Sparkle is insecure, just that the proof in [CKM23a] has a gap.

Other Algebraic Structures. In the pairing setting, a natural construction is the (non-interactive) threshold version of the BLS signature scheme [BLS01, Bol03], which has been modified to achieve adaptive security in [LJY14]. Recently, Bacho and Loss [BL22] have proven adaptive security of threshold BLS in the AGM. Das et al. have constructed weighted threshold signatures in the pairing-setting [DCX+23], and Crites et al. have constructed structure-preserving threshold signatures in the pairing-setting [CKP+23]. Threshold signatures have been constructed based on RSA [DDFY94, Rab98, FMY98, Sho00, ADN06, GHKR08, TZ23]. Notably, adaptive security has been considered in [ADN06]. A few works also have constructed threshold signatures from lattices [BKP13, BGG+18, DOTT21, ASY22, GKS23]. Finally, several works have proposed threshold signing protocols for ECDSA signatures [GGN16, LN18, GG18, DKLs19, DJN+20, GG20, CGG+20, CCL+20, GKSŚ20]. Except for [CGG+20], these works focus on static corruptions. For an overview of this line of work, see [AHS20].

Robustness. Recently, there has been renewed interest in robust (Schnorr) threshold signing protocols [RRJ+22, BHK+23, Sho23, GS23]. Such robust protocols additionally ensure that no malicious party can prevent honest parties from signing. Notably, all of these protocols assume static corruptions.

Multi-Signatures. Multi-signatures [IN83, BN06] are threshold signatures with $t = n - 1$, i.e., all n parties need to participate in the signing protocol, with the advantage that parties generate their keys independently and come together to sign spontaneously without setting up a shared key. There is a rich literature on multi-signatures, e.g., [Bol03, BDN18, MPSW19, NRSW20, NRS21, BD21, AB21, BTT22, FSZ22, TZ23]. We refer to Chapter 4 for more details.

Distributed Key Generation. In principle, one can rely on generic secure multi-party computation to set up key shares for a threshold signature scheme without using a trusted dealer. To get a more efficient solution, dedicated distributed key generation protocols have been studied [Ped92, CGJ+99, JL00, GJKR07, KMS20, DYX+22, KGS23], with some of them being adaptively secure [CGJ+99, JL00, KMS20].

5.1.3 Outline

In Section 5.2, we give a technical overview. Then, in Section 5.3, we define our formal model for threshold signatures, including syntax and security. In Section 5.4, we construct our threshold signature scheme generically from an abstract building block we call tagged linear function families. We present two instantiations of this abstract construction in Section 5.5. Finally, in Section 5.6, we discuss the efficiency of the resulting schemes.

5.2 Technical Overview

In this section, we outline our techniques in an informal way. We keep the overview self-contained, but some background on Schnorr signatures [Sch91, KMP16], five-move identification [Che05, GJKW07, KLP17], and Sparkle [CKM23a] is helpful.

Sparkle and The Problem with Rewinding. As our starting point, let us review the main ideas behind Sparkle [CKM23a], and why the use of rewinding limits us to tolerating at most $t/2$ corruptions. For that, we fix a group $\mathbb{G}$ with generator g and prime order p. Each signer $i \in [n]$ holds a secret key share $\mathsf{sk}_i \in \mathbb{Z}_p$ such that $\mathsf{sk}_i = f(i)$ for a polynomial f of degree t. Further, the public key is $\mathsf{pk} = g^{f(0)}$. To sign a message m, a set $S \subseteq [n]$ of signers engage in the following interactive signing protocol, omitting some details:

1. Each party $i \in S$ samples a random $r_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $R_i = g^{r_i}$. It then sends a hash com_i of R_i, S, and m to the other signers to commit to R_i. We call R_i a *preimage* of com_i. The hash function is modeled as a random oracle.
2. Once a party has received all hashes from the first round, it sends R_i to the other signers to open the commitment.

3. If all commitments are correctly opened, each signer computes the combined nonce $R = \prod_i R_i$. Then, it derives a challenge $c \in \mathbb{Z}_p$ from pk, R, and m using another random oracle. Each signer i computes and sends its response share $s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$, where $\ell_{i,S}$ is a Lagrange coefficient. The signature is (c, s), where $s = \sum_i s_i$.

The overall proof strategy adopted in [CKM23a] follows a similar paradigm as that of proving Schnorr signatures, with appropriate twists. Namely, one first takes care of simulating signing queries using honest-verifier zero-knowledge (HVZK) and by suitably programming the random oracle. We will come back to this part of the proof later. Then, via rewinding, one can extract the secret key from a forgery. To simulate adaptive corruption queries, the proof of Sparkle relies on a DLOG oracle on each corruption query, i.e., security is proven under the one-more version of DLOG (OMDL). Specifically, getting $t + 1$ DLOG challenges from the OMDL assumption and t-time access to a DLOG oracle, the reduction defines a degree t polynomial "in the exponent", simulates the game as explained, and uses rewinding to solve the final DLOG challenge. Note that if we allow the adversary to corrupt at most t_c parties throughout the experiment, it may corrupt up to $2t_c$ parties over both runs, meaning that the reduction has to query the DLOG oracle up to $2t_c$ times. Therefore, we have to require that $2t_c \leq t$.

How to Avoid Rewinding. Now it should be clear that the restriction on the corruption threshold is induced by the use of rewinding. If we avoid rewinding, we can also remove the restriction. To do so, it is natural to follow existing approaches from the literature on tightly-secure (and thus rewinding-free) signatures. A common approach is to rely on lossy identification [KW03, AFLT12, KMP16] that has already been used in the closely-related multi-signature setting, see Chapter 4. We find this unsuitable for two reasons. Namely, (a) these schemes rely on the DDH assumption, it is not clear at all what a suitable one-more variant would look like, and (b) the core idea of this technique is to move to a hybrid in which there is no secret key for pk at all. This seems hard to combine with adaptive corruptions. Roughly, this is because if there is no secret key for pk, then at most t of the pk_i can have a secret key, meaning that we would have to guess the set of corruptions. Instead, we take inspiration from five-move identification [Che05, GJKW07, KLP17], for which problems (a) and (b) do not show up. Namely, (a) such schemes rely on the CDH assumption, and (b) there is always a secret key. To explain the idea, we directly focus on our threshold signature scheme. For that, let $h \in \mathbb{G}$ be *derived from the message* via a random oracle. Given h, our signing protocol is as follows:

1. Each signer $i \in S$ samples $r_i \xleftarrow{\$} \mathbb{Z}_p$ and computes $R_i^{(1)} = g^{r_i}, R_i^{(2)} = h^{r_i}$, and $\mathsf{pk}_i^{(2)} = h^{\mathsf{sk}_i}$. It then sends a hash of $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$ to the other signers.

2. Once a party received all hashes from the first round, it sends $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$.

3. If all commitments are correctly opened, each signer computes the combined nonces $R^{(k)}$ for $k \in \{1, 2\}$ and secondary public key $\mathsf{pk}^{(2)}$ in a natural way. Then, it derives a challenge c from $R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}$, and m and computes $s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$. The signature is $(\mathsf{pk}^{(2)}, c, s)$ with $s = \sum_i s_i$.

Intuitively, the signers engage in two executions of Sparkle with generators g and h, respectively, using the same randomness r_i. To understand why we can avoid rewinding with this scheme, let us ignore signing and corruption queries for a moment, and focus on how to turn a forgery $(\mathsf{pk}^{(2)}, c, s)$ into a solution for a hard problem, concretely, CDH. For that, we consider two cases. First, if $\mathsf{pk}^{(2)} = h^{f(0)}$, then $\mathsf{pk}^{(2)}$ is a CDH solution for $\mathsf{pk} = g^{f(0)}$ and h. Indeed, this is what should happen in an honest execution. Second, we can bound the probability that the forgery is valid and $\mathsf{pk}^{(2)} \neq h^{f(0)}$ using a statistical argument. Roughly, (c, s) acts as a statistically sound proof for the statement $\mathsf{pk}^{(2)} = h^{f(0)}$. To simulate adaptive corruptions, for now assume that we can rely on a one-more variant of the CDH assumption, in which we have t-time access to a DLOG oracle. We come back to this later. What remains is to simulate honest parties during the signing. For that, the first trick is to set up h (by programming the random oracle) in a special way. Roughly, we want to be able to translate valid

transcripts with respect to g into valid transcripts with respect to h. Once this is established, we can focus on simulating the g-side of the protocol.

A Gap In the Proof of Sparkle. If we only focus on the g-side, our protocol is essentially Sparkle. Therefore, it should be possible to simulate signing exactly as in Sparkle using HVZK. Unfortunately, when looking at this part of Sparkle's proof, we discovered that a certain adversarial behavior is not covered. Namely, the proof does not correctly simulate the case in which the adversary sends inconsistent sets of commitments to different honest parties. It turns out that handling this requires fundamentally new techniques. To understand the gap, it is instructive to consider Sparkle's proof for an example of three signers in a session sid, with two of them being honest, say Signer 1 and 2, and the third one being malicious. Let us assume that Signers 1 and 2 are already in the second round of the protocol. That is, both already sent their commitments com_1 and com_2 and now expect a list of commitments $\mathcal{M} = (\mathsf{com}_1, \mathsf{com}_2, \mathsf{com}_3)$ from the first round as input. In Sparkle's proof, the reduction sends random commitments com_1 and com_2 on behalf of the honest parties. Later, when Signer 1 (resp. 2) gets $\mathcal{M}$, it has to output its second message R_1 (resp. R_2) and program the random oracle at R_1 (resp. R_2) to be com_1 (resp. com_2). The goal of the reduction is to set up R_1 and R_2 using HVZK such that the responses s_1 and s_2 can be computed without using the secret key. To understand how the reduction proceeds, assume that Signer 1 is asked (by the adversary) to reveal his nonce R_1 first. When this happens, the reduction samples a challenge c and a response s_1. It then defines R_1 as $R_1 := g^{s_1}\mathsf{pk}_1^{-c\ell_{1,S}}$. Ideally, the reduction would now program the random oracle on the combined nonce $R = R_1R_2R_3$ to return c, and output R_1 to the adversary. However, while the reduction can extract R_3 from com_3 by observing the random oracle queries, R_2 is not yet defined at that point. The solution proposed in Sparkle's proof is as follows. Before returning R_1 to the adversary, the reduction also samples s_2 and defines $R_2 := g^{s_2}\mathsf{pk}_2^{-c\ell_{2,S}}$. Then, the reduction can compute the combined nonce $R = R_1R_2R_3$ and program the random oracle on input R to return c. Later, it can use s_1 and s_2 as responses.

However, as we will argue now, this strategy is flawed[3]. Think about what happens if the first-round messages $\mathcal{M}'$ that Signer 2 sees do not contain com_3, but instead a different[4] commitment com_3' to a nonce $R_3' \neq R_3$. Then, with high probability, the combined nonce R' that Signer 2 will compute is different from R, meaning that its challenge c' will also be different from c, and so s_2 is not a valid response. One naive idea to solve this is to program $R_2 := g^{s_2}\mathsf{pk}_2^{-c'\ell_{2,S}}$ for an independent c' when we reveal R_1. In this case, however, the adversary may just choose to submit $\mathcal{M}' = \mathcal{M}$ to Signer 2, making the simulation fail.

Equivalence Classes to the Rescue. The solution we present is very technical, and we sketch a massively simplified solution here. Abstractly speaking, we want to be able to identify whether two queries $q = (sid, i, \mathcal{M})$ and $q' = (sid', i', \mathcal{M}')$ will result in the same combined nonce *before* all commitments com_j in $\mathcal{M}$ and $\mathcal{M}'$ have preimages R_j. To do so, we define an equivalence relation $\sim$ on such queries for which we show two properties.

1. First, the equivalence relation is consistent over time, namely, (a) if $q \sim q'$ at some point in time, then $q \sim q'$ at any later point, and (b) if $q \not\sim q'$ at some point in time, then $q \not\sim q'$ at any later point.

2. Second, assume that all commitments in $\mathcal{M}$ and $\mathcal{M}'$ have preimages. Then the resulting combined nonces R and R' are the same if and only if $q \sim q'$.

The technical challenge is that $\sim$ has to stay consistent while also adapting to changes in the random oracle over time. Assuming we have such a relation, we can make the simulation work. Namely, when

[3]The problem has nothing to do with adaptive security and shows up for a static adversary as well.

[4]Note that in Sparkle, no broadcast channel is assumed, and so this may happen. Also, note that in multi-signatures that follow a similar strategy, e.g., [BN06], this problem does not show up as there is only one honest signer.

we have to reveal the nonce R_i of an honest signer i, we first define $c := \mathsf{C}(q)$, where C is a *random oracle on equivalence classes* and is only known to the reduction. That is, C is a random oracle with the additional condition that $\mathsf{C}(q) = \mathsf{C}(q')$ if $q \sim q'$. Then, we define $R_i := g^{s_i}\mathsf{pk}_i^{-c\ell_{i,S}}$. We do not define any other $R_{i'}$ of honest parties at that point, meaning that we also may not know the combined nonce yet. Instead, we carefully delay the random oracle programming of the combined nonce until it is completely known.

Cherry on Top: Non-Interactive Assumptions. While the scheme we have so far does its job, we still rely on an interactive assumption, and we are eager to avoid it. For that, it is useful to write our scheme abstractly, replacing every exponentiation with the function $\mathsf{T}(t, x) = t^x$. Note that for almost every $t \in \mathbb{G}$, the function $\mathsf{T}(t, \cdot)$ is a bijection. Our hope is that by instantiating our scheme with a different function with suitable properties, we can show that the corresponding one-more assumption is implied by a non-interactive assumption. Indeed, Tessaro and Zhu [TZ23] recently used a similar strategy to avoid OMDL in certain situations. To do so, they replace the bijective function with a compressing function. In our case, the interactive assumption, written abstractly using T, asks an adversary to win the following game:

- A random g and h are sampled, and random $x_0, \ldots, x_t$ are sampled. Then, g, h, and all $X_i = \mathsf{T}(g, x_i)$ for all $0 \le i \le t$ are given to the adversary.
- Roughly, the adversary gets t-time access to an algebraic oracle inverting T. More precisely, the oracle outputs $\sum_{i=0}^{t} \alpha_i x_i$ on input $\alpha_0, \ldots, \alpha_t$.
- The adversary outputs X'_i for all $0 \le i \le t$. It wins if all solutions are valid, meaning that there is a z_i such that $\mathsf{T}(g, z_i) = X_i \wedge \mathsf{T}(h, z_i) = X'_i$. Intuitively, the adversary has to "shift" the images X_i from g to h.

Under a suitable instantiation of T and a well-studied non-interactive assumption, we want to show that no adversary can win this game. Unfortunately, if we just use a compressing function as in the case of [TZ23], it is not clear how to make use of the winning condition. Instead, our idea is to use a function that can *dynamically* be switched between a bijective and a compressing mode. A bit more precisely, a proof sketch works as follows:

1. We start with the game we introduced above. With overwhelming probability, the functions $\mathsf{T}_g := \mathsf{T}(g, \cdot)$ and $\mathsf{T}_h := \mathsf{T}(h, \cdot)$ should be bijective.
2. Assume that we can efficiently invert T_h using knowledge of h. Then, we can state our winning condition equivalently by requiring that $\mathsf{T}_h^{-1}(X'_i) = x_i$ for all i. Roughly, this means that the adversary has to find the x_i to win.
3. We assume that we can indistinguishably switch g to a mode in which T_g is compressing.
4. Finally, we use a statistical argument to show that the adversary can not win. Intuitively, this is because T_g is compressing and the inversion oracle does not leak too much about the x_i's.

It turns out that, choosing T carefully, we find a function that (1) has all the properties we need for our scheme and (2) allows us to follow our proof sketch under the DDH assumption.

5.3 Preliminaries for this Chapter

In this section, we define threshold signatures, the main object of study for this chapter.

Threshold Signatures. We define threshold signatures assuming a trusted key generation, which can be replaced by a distributed key generation in practice. Our syntax matches the three-round structure of our protocol. Namely, a (t, n)-threshold signature scheme is a tuple of PPT algorithms $\mathsf{TS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, where $\mathsf{Setup}(1^\lambda)$ outputs system parameters par, and $\mathsf{Gen}(\mathsf{par})$ outputs a public key pk

and secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$. Further, Sig specifies a signing protocol, formally split into four algorithms $(\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2, \mathsf{Combine})$. Here, algorithm Sig_j models how the signers locally compute their $(j+1)$st protocol message pm_{j+1} and advance their state, where $\mathsf{Sig}_0(S, i, \mathsf{sk}_i, \mathsf{m})$ takes as input the signer set S, the index of the signer $i \in [n]$, its secret key share sk_i, and the message m, and Sig_1 (resp. Sig_2) takes as input the current state of the signer and the list $\mathcal{M}_1$ (resp $\mathcal{M}_2$) of all protocol messages from the previous round. Finally, $\mathsf{Combine}(S, \mathsf{m}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$ can be used to publicly turn the transcript into a signature σ, which can then be verified using $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$. Roughly, we say that the scheme is complete if for any such parameters and keys, a signature generated by a signing protocol among $t+1$ parties outputs a signature for which Ver outputs 1.

Definition 5.1 (Threshold Signature Scheme)**.** *Let $t < n$ be natural numbers. A (t, n)-threshold signature scheme is a tuple of PPT algorithms $\mathsf{TS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter 1^λ and outputs global system parameters* par*, where* par *implicitly defines sets of public keys, secret keys, messages and signatures, and all algorithms related to* TS *implicitly take* par *as input.*
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ *takes as input system parameters* par*, and outputs a public key* pk *and secret key shares $\mathsf{sk}_1, \ldots, \mathsf{sk}_n$.*
- $\mathsf{Sig} = (\mathsf{Sig}_0, \mathsf{Sig}_1, \mathsf{Sig}_2, \mathsf{Combine})$ *is split into four algorithms:*
  - $\mathsf{Sig}_0(S, i, \mathsf{sk}_i, \mathsf{m}) \to (\mathsf{pm}_1, St_1)$ *takes as input a signer set $S \subseteq [n]$, an index $i \in [n]$, a secret key share sk_i, and a message* m*, and outputs a protocol message pm_1 and a state St_1.*
  - $\mathsf{Sig}_1(St_1, \mathcal{M}_1) \to (\mathsf{pm}_2, St_2)$ *takes as input a state St_1 and a tuple $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,l})$ of protocol messages, and outputs a protocol message pm_2 and a state St_2.*
  - $\mathsf{Sig}_2(St_2, \mathcal{M}_2) \to \mathsf{pm}_3$ *takes as input a state St_2 and a tuple $\mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,l})$ of protocol messages, and outputs a protocol message pm_3.*
  - $\mathsf{Combine}(S, \mathsf{m}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3) \to \sigma$ *takes as input a signer set $S \subseteq [n]$, a message* m*, tuples $\mathcal{M}_1 = (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,l}), \mathcal{M}_2 = (\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,l})$, and $\mathcal{M}_3 = (\mathsf{pm}_{3,1}, \ldots, \mathsf{pm}_{3,l})$ of protocol messages, and outputs a signature σ.*
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input a public key* pk*, a message* m*, and a signature σ, and outputs a bit $b \in \{0, 1\}$.*

We require that TS *is complete in the following sense. For all* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \in \mathsf{Gen}(\mathsf{par})$*, all messages* m*, and all $S \subseteq [n]$ with $|S| = t+1$ we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) = 1 \;\middle|\; \sigma \leftarrow \mathsf{TS.Exec}(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, S, \mathsf{m})\right] = 1,$$

where algorithm TS.Exec *is defined in Figure 5.1.*

Our security game is in line with the established template and is presented in Figure 5.2. First, the adversary gets an honestly generated public key as input. At any point in time, the adversary can start a new signing session with signer set S and message m with session identifier sid by calling an oracle $\textsc{Next}(sid, S, \mathsf{m})$. Additionally, the adversary may adaptively corrupt up to t users via an oracle Corr. Thereby, it learns their secret key and private state in all currently open signing sessions. To interact with honest users in signing sessions, the adversary has access to per-round signing oracles $\textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2$. Roughly, each signing oracle can be called with respect to a specific honest user i and a session identifier sid, given that the user is already in the respective round for that session (modeled by algorithm Allowed). Further, when calling such an oracle, the adversary inputs the vector of all messages of the previous round. In particular, the adversary could send different messages to two different honest parties within the same session, i.e., we assume no broadcast channels. Additionally, this means that the adversary can arbitrarily decide which message to send to an honest party on behalf

Alg $\mathsf{TS.Exec}(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n, S, \mathsf{m})$

01 **if** $|S| \neq t+1 \vee S \not\subseteq [n]$: **return** $\perp$
02 **parse** $\{i_1, \ldots, i_{t+1}\} := S$ s.t. $i_1 < i_2 \cdots i_t < i_{t+1}$
03 **for** $j \in [t+1]$: $(\mathsf{pm}_{1,i_j}, St_{1,i_j}) \leftarrow \mathsf{Sig}_0(S, i_j, \mathsf{sk}_{i_j}, \mathsf{m})$
04 $\mathcal{M}_1 := (\mathsf{pm}_{1,i_1}, \ldots, \mathsf{pm}_{1,i_{t+1}})$
05 **for** $j \in [t+1]$: $(\mathsf{pm}_{2,i_j}, St_{2,i_j}) \leftarrow \mathsf{Sig}_1(St_{1,i_j}, \mathcal{M}_1)$
06 $\mathcal{M}_2 := (\mathsf{pm}_{2,i_1}, \ldots, \mathsf{pm}_{2,i_{t+1}})$
07 **for** $j \in [t+1]$: $\mathsf{pm}_{3,i_j} \leftarrow \mathsf{Sig}_2(St_{2,i_j}, \mathcal{M}_2)$
08 $\mathcal{M}_3 := (\mathsf{pm}_{3,i_1}, \ldots, \mathsf{pm}_{3,i_{t+1}})$
09 **return** $\sigma \leftarrow \mathsf{Combine}(S, \mathsf{m}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$

Figure 5.1: Algorithm TS.Exec for a (t, n)-threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. The algorithm models an honest execution of the signing protocol Sig.

of another honest party, i.e., we assume no authenticated channels. Finally, the adversary outputs a forgery (m^*, σ^*). It wins the security game, if it never started a signing session for message m^* and the signature σ^* is valid. Therefore, our notion is (an interactive version of) TS-UF-0 using the terminology of [BTZ22, BCK+22], which is similar to recent works [CKM23a, CGRS23].

No Erasures. In our pseudocode, the private state of signer i in session sid is stored in $\mathsf{state}[sid, i]$, where state is a map. After each signing round, this state is updated. We choose to update the state instead of adding a new state to avoid clutter, which is similar to earlier works [CKM23a]. On the downside, this means that potentially, schemes that are secure in our model could rely on erasures, i.e., on safely deleting part of the state of an earlier round before a user gets corrupted. We emphasize that in our scheme, any state in earlier rounds can be computed from the state in the current round and the secret key. This means that our schemes do not rely on erasures.

Definition 5.2 (TS-EUF-CMA Security)**.** *Let* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a* (t, n)*-threshold signature scheme. Consider the game* $\mathbf{TS\text{-}EUF\text{-}CMA}$ *defined in Figure 5.2. We say that* TS *is* TS-EUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{TS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) := \Pr\left[\mathbf{TS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}(\lambda) \Rightarrow 1\right].$$

5.4 Abstract Construction

In this section, we present our threshold signature scheme in an abstract way. Namely, we first introduce a building block we call tagged linear function families. Then, we show how to construct threshold signatures from tagged linear function families.

5.4.1 Building Block: Tagged Linear Function Families

Similar to what is done in other works [HKL19, KLR21, TZ23] and Chapter 4, we use the abstraction of linear function families to describe our scheme in a generic way. However, we slightly change the notion by introducing tags to cover different functions with the same set of parameters.

Definition 5.3 (Tagged Linear Function Family)**.** *A tagged linear function family (TLFF) is a tuple of PPT algorithms* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs parameters* par*. We assume that* par *implicitly defines the following sets: A set of scalars* $\mathcal{S}_{\mathsf{par}}$*, which forms a field; a set of tags* $\mathcal{T}_{\mathsf{par}}$*; a domain* $\mathcal{D}_{\mathsf{par}}$ *and a range* $\mathcal{R}_{\mathsf{par}}$*, where each forms a vector space over* $\mathcal{S}_{\mathsf{par}}$*. If* par *is clear*

Game TS-EUF-CMA$^{\mathcal{A}}_{\mathsf{TS}}(\lambda)$

$\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n) \leftarrow \mathsf{Gen}(\mathsf{par})$
$\textsc{Sig} := (\textsc{Next}, \textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2)$
$(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sig},\textsc{Corr}}(\mathsf{par}, \mathsf{pk})$
if $\mathsf{m}^* \in \mathsf{Queried}$: **return** 0
return $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}^*, \sigma^*)$

Oracle $\textsc{Corr}(i)$

if $|\mathsf{Corrupted}| \geq t$: **return** $\bot$
$\mathsf{Corrupted} := \mathsf{Corrupted} \cup \{i\}$
return $(\mathsf{sk}_i, \mathsf{state}[\cdot, i])$

Oracle $\textsc{Next}(sid, S, \mathsf{m})$

if $|S| \neq t+1 \vee S \not\subseteq [n]$: **return** $\bot$
if $sid \in \mathsf{Sessions}$: **return** $\bot$
$\mathsf{Sessions} := \mathsf{Sessions} \cup \{sid\}$
$\mathsf{message}[sid] := \mathsf{m},\ \mathsf{signers}[sid] := S$
$\mathsf{Queried} := \mathsf{Queried} \cup \{\mathsf{m}\}$
for $i \in S$: $\mathsf{round}[sid, i] := 0$

Oracle $\textsc{Sig}_0(sid, i)$

if $\mathsf{Allowed}(sid, i, 0, \bot) = 0$:
  return $\bot$
$S := \mathsf{signers}[sid]$
$(\mathsf{pm}, St) \leftarrow \mathsf{Sig}_0(S, i, \mathsf{sk}_i, \mathsf{m})$
$\mathsf{pm}_1[sid, i] := \mathsf{pm},\ \mathsf{state}[sid, i] := St$
$\mathsf{round}[sid, i] := 1$
return pm

Oracle $\textsc{Sig}_1(sid, i, \mathcal{M}_1)$

if $\mathsf{Allowed}(sid, i, 1, \mathcal{M}_1) = 0$:
  return $\bot$
$(\mathsf{pm}, St) \leftarrow \mathsf{Sig}_1(\mathsf{state}[sid, i], \mathcal{M}_1)$
$\mathsf{pm}_2[sid, i] := \mathsf{pm},\ \mathsf{state}[sid, i] := St$
$\mathsf{round}[sid, i] := 2$
return pm

Oracle $\textsc{Sig}_2(sid, i, \mathcal{M}_2)$

if $\mathsf{Allowed}(sid, i, 2, \mathcal{M}_2) = 0$:
  return $\bot$
$\mathsf{pm} \leftarrow \mathsf{Sig}_2(\mathsf{state}[sid, i], \mathcal{M}_2)$
$\mathsf{round}[sid, i] := 3$
return pm

Alg $\mathsf{Allowed}(sid, i, r, \mathcal{M})$

if $sid \notin \mathsf{Sessions}$: **return** 0
$S := \mathsf{signers}[sid],\ H := S \setminus \mathsf{Corrupted}$
if $i \notin H$: **return** 0
if $\mathsf{round}[sid, i] \neq r$: **return** 0
if $r > 0$:
  parse $(\mathsf{pm}_i)_{i \in S} := \mathcal{M}$
  if $\mathsf{pm}_i \neq \mathsf{pm}_r[sid, i]$: **return** 0
return 1

Figure 5.2: The game **TS-EUF-CMA** for a (three-round) (t, n)-threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ and an adversary $\mathcal{A}$.

from the context, we omit the subscript par*. We naturally denote the operations of these fields and vector spaces by $+$ and $\cdot$, and assume that these operations can be evaluated efficiently.*

- $\mathsf{T}(\mathsf{par}, g, x) \to X$ *is deterministic, takes as input parameters* par*, a tag $g \in \mathcal{T}$, a domain element $x \in \mathcal{D}$, and outputs a range element $X \in \mathcal{R}$. For all parameters* par*, and for all tags $g \in \mathcal{T}$, the function* $\mathsf{T}(\mathsf{par}, g, \cdot)$ *realizes a homomorphism, i.e.*

$$\forall s \in \mathcal{S}, x, y \in \mathcal{D} : \ \mathsf{T}(\mathsf{par}, g, s \cdot x + y) = s \cdot \mathsf{T}(\mathsf{par}, g, x) + \mathsf{T}(\mathsf{par}, g, y).$$

For T*, we also omit the input* par *if it is clear from the context.*

For our construction, we require that images are uniformly distributed. More precisely, we say that TLF is ε_{r}-regular, if there is a set Reg of pairs (par, g) such that random parameters par and tags g are in Reg with probability at least $1 - \varepsilon_{\mathsf{r}}$, and for each such pair in Reg, $\mathsf{T}(\mathsf{par}, g, x)$ is uniformly distributed over the range, assuming $x \xleftarrow{\$} \mathcal{D}$. We give the formal definition next.

Definition 5.4 (Regular TLFF)**.** *Let* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *be a tagged linear function family. We say that* TLF *is ε_{r}-regular, if there is a set* Reg *such that the following two properties hold:*

- *We have*

$$\Pr\left[(\mathsf{par}, g) \notin \mathsf{Reg} \mid \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ g \xleftarrow{\$} \mathcal{T}\right] \leq \varepsilon_{\mathsf{r}}.$$

- *For any fixed* $(\mathsf{par}, g) \in \mathsf{Reg}$*, the following distributions are the same:*

$$\{(\mathsf{par}, g, X) \mid X \xleftarrow{\$} \mathcal{R}\} \text{ and } \{(\mathsf{par}, g, X) \mid x \xleftarrow{\$} \mathcal{D},\ X := \mathsf{T}(\mathsf{par}, g, x)\}\,.$$

Next, we show that tagged linear function families satisfy a statistical property that turns out to be useful. This property is implicitly present in other works as well, e.g., in [KW03, AFLT12, KLP17] or Chapter 4, and can be interpreted in various ways, e.g., as the soundness of a natural proof system.

Lemma 5.1. *Let* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *be a tagged linear function family. For every fixed parameters* par *and tags* $g, h \in \mathcal{T}$*, define the set*

$$\mathsf{Im}(\mathsf{par}, g, h) := \left\{(X_1, X_2) \in \mathcal{R}^2 \;\middle|\; \exists x \in \mathcal{D} :\ \mathsf{T}(g, x) = X_1 \wedge \mathsf{T}(h, x) = X_2\right\}.$$

Then, for any (even unbounded) algorithm $\mathcal{A}$*, we have*

$$\Pr\left[\begin{array}{lll} & (X_1, X_2) \notin \mathsf{Im}(\mathsf{par}, g, h) & \\ \wedge & \mathsf{T}(g, s) = c \cdot X_1 + R_1 & \\ \wedge & \mathsf{T}(h, s) = c \cdot X_2 + R_2 & \end{array} \;\middle|\; \begin{array}{l} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda), \\ (St, g, h, X_1, X_2, R_1, R_2) \leftarrow \mathcal{A}(\mathsf{par}), \\ c \xleftarrow{\$} \mathcal{S},\ \ s \leftarrow \mathcal{A}(St, c) \end{array}\right] \leq \frac{1}{|\mathcal{S}|}.$$

Proof. We claim that for each fixed values $\mathsf{par}, g, h, X_1, X_2, R_1, R_2$ such that $(X_1, X_2) \notin \mathsf{Im}(\mathsf{par}, g, h)$, the following set contains at most one element:

$$\mathsf{BadC} := \{c \in \mathcal{S} \mid \exists s \in \mathcal{D} :\ \mathsf{T}(g, s) = c \cdot X_1 + R_1 \wedge \mathsf{T}(h, s) = c \cdot X_2 + R_2\}.$$

The reader may observe that this is indeed sufficient to show the lemma. To show the claim, assume towards contradiction that there are two distinct $c \neq c'$ in BadC and $s, s' \in \mathcal{D}$ were the corresponding witnesses with

$$\begin{array}{rll} & \mathsf{T}(g, s) = c \cdot X_1 + R_1, & \mathsf{T}(g, s') = c' \cdot X_1 + R_1 \\ \text{and} & \mathsf{T}(h, s) = c \cdot X_2 + R_2, & \mathsf{T}(h, s') = c' \cdot X_2 + R_2. \end{array}$$

We rearrange these equations and get

$$\begin{aligned} \mathsf{T}(g, s) - c \cdot X_1 = R_1 = \mathsf{T}(g, s') - c' \cdot X_1 \\ \text{and } \mathsf{T}(h, s) - c \cdot X_2 = R_2 = \mathsf{T}(h, s') - c' \cdot X_2. \end{aligned}$$

Rearranging again, using the linearity of T for any fixed tag, and solving for (X_1, X_2), we get

$$X_1 = \mathsf{T}\left(g, \frac{s - s'}{c - c'}\right) \text{ and } X_2 = \mathsf{T}\left(h, \frac{s - s'}{c - c'}\right).$$

Hence, $(X_1, X_2) \in \mathsf{Im}(\mathsf{par}, t, h)$. With this contradiction, we conclude. □

As another technical tool in our proof, we need our tagged linear function families to be translatable, a notion we define next. Informally, it means that we can rerandomize a given tag g into a tag h, such that we can efficiently compute $\mathsf{T}(h, x)$ from $\mathsf{T}(g, x)$ without knowing x.

Definition 5.5 (Translatability)**.** *Let* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *be a tagged linear function family. We say that* TLF *is* ε_{t}*-translatable, if there is a PPT algorithm* Shift *and a deterministic polynomial time algorithm* $\mathsf{Translate}$*, such that the following properties hold:*

- ***Well Distributed Tags.*** *The statistical distance between the following distributions* $\mathcal{X}_0$ *and* $\mathcal{X}_1$ *is at most* ε_{t}*:*

$$\begin{aligned} \mathcal{X}_0 &:= \left\{(\mathsf{par}, g, h) \;\middle|\; \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ g \xleftarrow{\$} \mathcal{T},\ h \xleftarrow{\$} \mathcal{T}\right\}, \\ \mathcal{X}_1 &:= \left\{(\mathsf{par}, g, h) \;\middle|\; \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ g \xleftarrow{\$} \mathcal{T},\ (h, \mathsf{td}) \leftarrow \mathsf{Shift}(\mathsf{par}, g)\right\}. \end{aligned}$$

- ***Translation Completeness.*** *For every* $\mathsf{par} \in \mathsf{Gen}(1^\lambda)$*, for any* $g \in \mathcal{T}$*, any* $x \in \mathcal{D}$*, and any* $(h, \mathsf{td}) \in \mathsf{Shift}(\mathsf{par}, g)$*, we have*

$$\mathsf{Translate}(\mathsf{td}, \mathsf{T}(g,x)) = \mathsf{T}(h,x) \text{ and } \mathsf{InvTranslate}(\mathsf{td}, \mathsf{T}(h,x)) = \mathsf{T}(g,x).$$

Next, we define the main security property that we will require for our construction. Intuitively, it should not be possible for an adversary to translate $\mathsf{T}(g,x)$ into $\mathsf{T}(h,x)$ if g, h and x are chosen randomly. Our actual notion is a one-more variant of this intuition.

Definition 5.6 (Algebraic Translation Resistance)**.** *Let* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *be a tagged linear function family, and* $t \in \mathbb{N}$ *be a number. Consider the game* **A-TRAN-RES** *defined in Figure 5.3. We say that* TLF *is* t*-algebraic translation resistant, if for any PPT algorithm* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{t\text{-A-TRAN-RES}}_{\mathcal{A},\mathsf{TLF}}(\lambda) := \Pr\left[t\text{-}\mathbf{A\text{-}TRAN\text{-}RES}^{\mathcal{A}}_{\mathsf{TLF}}(\lambda) \Rightarrow 1\right].$$

Game t**-A-TRAN-RES**$^{\mathcal{A}}_{\mathsf{TLF}}(\lambda)$

01 $\mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda),\ \ g, h \xleftarrow{\$} \mathcal{T},\ \ x_0, \ldots, x_t \xleftarrow{\$} \mathcal{D}$
02 **for** $i \in \{0\} \cup [t]:\ X_i := \mathsf{T}(g, x_i)$
03 $(X'_i)_{i=0}^t \leftarrow \mathcal{A}^{\text{Inv}}(\mathsf{par}, g, h, (X_i)_{i=0}^t)$
04 **if** $\forall i \in \{0\} \cup [t]\ \exists z \in \mathcal{D}$
s.t. $\mathsf{T}(g,z) = X_i \wedge \mathsf{T}(h,z) = X'_i:$ **return** 1
05 **return** 0

Oracle $\text{Inv}(\alpha_0, \ldots, \alpha_t)$

06 **if** $q \geq t:$ **return** $\perp$
07 $q := q + 1$
08 $x := \sum_{i=0}^t \alpha_i x_i$
09 **return** x

Figure 5.3: The game **A-TRAN-RES** for a tagged linear function family $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ and an algorithm $\mathcal{A}$.

5.4.2 Construction

Let $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ be a tagged linear function family. Further, let $\mathsf{H}\colon \{0,1\}^* \to \mathcal{T}$, $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$, $\bar{\mathsf{H}}\colon \{0,1\}^* \to \mathcal{S}$ be random oracles. We construct a (t,n)-treshold signature scheme $\mathsf{Twinkle}[\mathsf{TLF}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$. We assume that there is an implicit injection from $[n]$ into $\mathcal{S}$. Further, let $\ell_{i,S}(x) := \prod_{j \in S \setminus \{i\}} (j-x)/(j-i) \in \mathcal{S}$ denote the ith lagrange coefficient for all $i \in [n]$ and $S \subseteq [n]$, and let $\ell_{i,S} := \ell_{i,S}(0)$. We describe our scheme verbally. For completeness, we provide pseudocode in Figure A.12.

Setup and Key Generation. All parties have access to public parameters $\mathsf{par} \leftarrow \mathsf{TLF.Gen}(1^\lambda)$ which define the function T, and sets $\mathcal{S}, \mathcal{T}, \mathcal{D}$, and $\mathcal{R}$, and to a random tag $g \xleftarrow{\$} \mathcal{T}$. To generate keys, elements $a_j \xleftarrow{\$} \mathcal{D}$ for $j \in \{0\} \cup [t]$ are sampled. These elements form the coefficients of a polynomial of degree t. For each $i \in [n]$, we define the key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the ith signer as

$$\mathsf{sk}_i := \sum_{j=0}^{t} a_j i^j, \quad \mathsf{pk}_i := \mathsf{T}(g, \mathsf{sk}_i).$$

The shared public key is defined as $\mathsf{pk} := \mathsf{pk}_0 := \mathsf{T}(g, a_0)$.

Signing Protocol. Let $S \subseteq [n]$ be a set of signers of size $t+1$. We assume all signers are aware of the set S and a message $\mathsf{m} \in \{0,1\}^*$ to be signed. First, they all compute $h := \mathsf{H}(\mathsf{m})$. Then, they run the following protocol phases to compute the signature:

1. *Commitment Phase.* Each signer $i \in S$ samples $r_i \xleftarrow{\$} \mathcal{D}$ and computes

$$R_i^{(1)} := \mathsf{T}(g, r_i), \quad R_i^{(2)} := \mathsf{T}(h, r_i), \quad \mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i).$$

Then, each signer $i \in S$ computes a commitment

$$\mathsf{com}_i := \hat{\mathsf{H}}(S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$$

and sends com_i to the other signers.

2. *Opening Phase.* Each signer $i \in S$ sends $R_i^{(1)}, R_i^{(2)}$ and $\mathsf{pk}_i^{(2)}$ to all other signers.

3. *Response Phase.* Each signer $i \in S$ checks that $\mathsf{com}_j = \hat{\mathsf{H}}(S, j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)})$ holds for all $j \in S$. If one of these equations does not hold, the signer aborts. Otherwise, the signer defines

$$R^{(1)} := \sum_{j \in S} R_j^{(1)}, \quad R^{(2)} := \sum_{j \in S} R_j^{(2)}, \quad \mathsf{pk}^{(2)} := \sum_{j \in S} \ell_{j,S} \mathsf{pk}_j^{(2)}.$$

The signer computes $c := \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$ and $s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$. It sends s_i to all other signers.

The signature is $\sigma := (\mathsf{pk}^{(2)}, c, s)$ for $s := \sum_{j \in S} s_j$.

Verification. Let pk be a public key, let $\mathsf{m} \in \{0,1\}^*$ be a message and let $\sigma = (\mathsf{pk}^{(2)}, c, s)$ be a signature. To verify σ with respect to pk and m, one first computes $h := \mathsf{H}(\mathsf{m})$ and $R^{(1)} := \mathsf{T}(g, s) - c \cdot \mathsf{pk}$, $R^{(2)} := \mathsf{T}(h, s) - c \cdot \mathsf{pk}^{(2)}$. Then, one accepts the signature, i.e., outputs 1, if and only if $c = \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$.

Theorem 5.1. *Let* $\mathsf{TLF} = (\mathsf{Gen}, \mathsf{T})$ *be a tagged linear function family and let* $\mathsf{H}\colon \{0,1\}^* \to \mathcal{T}$, $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$, $\bar{\mathsf{H}}\colon \{0,1\}^* \to \mathcal{S}$ *be random oracles. Assume that* TLF *is* ε_{r}*-regular and* ε_{t}*-translatable. Further, assume that* TLF *is* t*-algebraic translation resistant. Then,* $\mathsf{Twinkle}[\mathsf{TLF}]$ *is* $\mathsf{TS\text{-}EUF\text{-}CMA}$ *secure.*

Concretely, for any PPT algorithm $\mathcal{A}$ *that makes at most* Q_S *queries in total to oracles* $\textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2$ *and at most* $Q_{\mathsf{H}}, Q_{\hat{\mathsf{H}}}, Q_{\bar{\mathsf{H}}}$ *queries to oracles* $\mathsf{H}, \hat{\mathsf{H}}, \bar{\mathsf{H}}$*, respectively, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{Twinkle}[\mathsf{TLF}]}(\lambda) \leq\ & 8Q_S Q_{\mathsf{H}} \varepsilon_{\mathsf{t}} + 8Q_S \varepsilon_{\mathsf{r}} + \frac{8Q_S^2 Q_{\hat{\mathsf{H}}} + 4Q_S^3(Q_S + t)}{|\mathcal{R}|} + \frac{4Q_S Q_{\bar{\mathsf{H}}}}{|\mathcal{S}|} \\ & + \frac{4Q_S^3 + 4Q_S Q_{\hat{\mathsf{H}}}^2 + 4Q_{\hat{\mathsf{H}}} Q_S^2 (t+1)}{2^{2\lambda}} + 4Q_S \cdot \mathsf{Adv}^{t\text{-A-TRAN-RES}}_{\mathcal{B},\mathsf{TLF}}(\lambda).\end{aligned}$$

Proof. Fix an adversary $\mathcal{A}$ against the security of $\mathsf{TS} := \mathsf{Twinkle}[\mathsf{TLF}]$. We prove the statement by presenting a sequence of games $\mathbf{G}_0$-$\mathbf{G}_8$. To accompany the verbal description, all games and associated oracles and algorithms are presented as pseudocode in Figures A.13 to A.18.

Game $\mathbf{G}_0$: This game is the security game $\mathbf{TS\text{-}EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{TS}}$ for threshold signatures. We recall the game to fix some notation. First, the game samples parameters par' for TLF and a tag $g \xleftarrow{\$} \mathcal{T}$. It also samples random coefficients $a_0, \ldots, a_t \xleftarrow{\$} \mathcal{D}$ and computes a public key $\mathsf{pk} := \mathsf{pk}_0 := \mathsf{T}(g, a_0)$ and secret key shares $\mathsf{sk}_i := \sum_{j=0}^{t} a_j i^j$ for each $i \in [n]$. For convenience, denote the corresponding public key shares by $\mathsf{pk}_i := \mathsf{T}(g, \mathsf{sk}_i)$. Then, the game runs $\mathcal{A}$ on input $\mathsf{par} := (\mathsf{par}', g)$ and pk with access to signing oracles, corruption oracles, and random oracles. Concretely, it gets access to random oracles H, $\hat{\mathsf{H}}$, and $\bar{\mathsf{H}}$, which are provided by the game in the standard lazy way using maps $h[\cdot]$, $\hat{h}[\cdot]$, and $\bar{h}[\cdot]$, respectively. The set of corrupted parties is denoted by $\mathsf{Corrupted}$ and the set of queried messages is denoted by $\mathsf{Queried}$. Finally, the adversary outputs a forgery (m^*, σ^*) and the game outputs 1 if $\mathsf{m}^* \notin \mathsf{Queried}$, $|\mathsf{Corrupted}| \leq t$, and σ^* is a valid signature for m^*. We make three purely conceptual changes to the game. First, we will never keep the secret key share sk_i explicitly in the states $\mathsf{state}[sid, i]$ for users i in a session sid, although the scheme description would require this. This is without loss of

generality, as the adversary only gets to see the states when it corrupts a user, and in this case it also gets sk_i. Second, we assume the adversary always queried $\mathsf{H}(\mathsf{m}^*)$ before outputting its forgery. Third, we assume that the adversary makes exactly t (distinct) corruption queries. These changes are without loss of generality and do not change the advantage of $\mathcal{A}$. Formally, one could build a wrapper adversary that internally runs $\mathcal{A}$, but makes a query $\mathsf{H}(\mathsf{m}^*)$ and enough corruption queries before terminating, and on every corruption query includes sk_i in the states before passing the result back to $\mathcal{A}$. Clearly, we have

$$\mathsf{Adv}^{\mathsf{TS\text{-}EUF\text{-}CMA}}_{\mathcal{A},\mathsf{TS}}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

The remainder of our proof is split into three parts. In the first part ($\mathbf{G}_1$-$\mathbf{G}_3$), we ensure that the game no longer needs secret key shares sk_i to compute $\mathsf{pk}_i^{(2)}$ in the signing oracle. Roughly, this is done by embedding shifted tags $(h, \mathsf{td}) \leftarrow \mathsf{Shift}(\mathsf{par}', g)$ into random oracle H for signing queries, and keeping random tags h for the query related to the forgery. In the second part ($\mathbf{G}_4$-$\mathbf{G}_{11}$), we use careful delayed random oracle programming, observability of the random oracle, and an honest-verifier zero-knowledge-style programming to simulate the remaining parts of the signing queries without sk_i. As a result, sk_i is only needed when the adversary corrupts users. In the third part, we analyze $\mathbf{G}_{11}$. This is done by distinguishing two cases. One of the cases is bounded using a statistical argument. The other case is bounded using a reduction breaking the t-algebraic translation resistance of TLF. We now proceed with the details.

Game $\mathbf{G}_1$: In this game, we introduce a map $b[\cdot]$ that maps messages m to bits $b[\mathsf{m}] \in \{0,1\}$. Concretely, whenever a query $\mathsf{H}(\mathsf{m})$ is made for which the hash value is not yet defined, the game samples $b[\mathsf{m}]$ from a Bernoulli distribution $\mathcal{B}_\gamma$ with parameter $\gamma = 1/(Q_S+1)$. That is, $b[\mathsf{m}]$ is set to 1 with probability $1/(Q_S+1)$ and to 0 otherwise. The game aborts if $b[\mathsf{m}] = 1$ for some message m for which the signing oracle is called, or $b[\mathsf{m}^*] = 0$ for the forgery message m^*. Clearly, if no abort occurs, games $\mathbf{G}_0$ and $\mathbf{G}_1$ are the same. Further the view of $\mathcal{A}$ is independent of the map b. We obtain

$$\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] = \gamma\left(1-\gamma\right)^{Q_S} \cdot \Pr\left[\mathbf{G}_0 \Rightarrow 1\right]$$

Now, we can use the fact $(1-1/x)^x \geq 1/4$ for all $x \geq 2$ and get

$$\gamma\left(1-\gamma\right)^{Q_S} = \frac{1}{Q_S+1}\left(1-\frac{1}{Q_S+1}\right)^{Q_S} = \frac{1}{Q_S}\left(1-\frac{1}{Q_S+1}\right)^{Q_S+1} \geq \frac{1}{4Q_S},$$

where the second equality follows from

$$\frac{1}{Q_S}\left(1-\frac{1}{Q_S+1}\right) = \frac{1}{Q_S} - \frac{1}{Q_S(Q_S+1)} = \frac{(Q_S+1)-1}{Q_S(Q_S+1)} = \frac{Q_S}{Q_S(Q_S+1)} = \frac{1}{Q_S+1}.$$

In combination, we get

$$\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] \geq \frac{1}{4Q_S} \cdot \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_2$: In game $\mathbf{G}_2$, we change the way queries to random oracle H are answered. Namely, for a query $\mathsf{H}(\mathsf{m})$ for which the hash value $h[\mathsf{m}]$ is not yet defined, the game samples $h[\mathsf{m}] \xleftarrow{\$} \mathcal{T}$ as a random tag exactly as the previous game did. However, now, if $b[\mathsf{m}] = 0$, the game samples $(h, \mathsf{td}) \leftarrow \mathsf{Shift}(\mathsf{par}', g)$ and sets $h[\mathsf{m}] := h$. Further, it stores td in a map tr as $tr[\mathsf{m}] := \mathsf{td}$. Clearly, $\mathbf{G}_1$ and $\mathbf{G}_2$ are indistinguishable by the ε_{t}-translatability of TLF. Concretely, one can easily see that

$$\left|\Pr\left[\mathbf{G}_1 \Rightarrow 1\right] - \Pr\left[\mathbf{G}_2 \Rightarrow 1\right]\right| \leq Q_{\mathsf{H}}\varepsilon_{\mathsf{t}}.$$

Game $\mathbf{G}_3$: In this game, we change how the values $\mathsf{pk}_i^{(2)}$ are computed by the signing oracle. To recall, in the commitment phase of the signing protocol, the signing oracle for user $i \in [n]$ in $\mathbf{G}_2$ would compute the value $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$, where $h = \mathsf{H}(\mathsf{m})$ and m is the message to be signed.

Also, the value $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$ is recomputed in the opening phase of the signing protocol and included in the output sent to the adversary. From $\mathbf{G}_3$ on, $\mathsf{pk}_i^{(2)}$ is computed differently, namely, as $\mathsf{pk}_i^{(2)} := \mathsf{Translate}(tr[\mathsf{m}], \mathsf{pk}_i)$. Observe that if the game did not abort, we know that $b[\mathsf{m}] = 0$ (see $\mathbf{G}_1$) and therefore h has been generated as $(h, \mathsf{td}) \leftarrow \mathsf{Shift}(\mathsf{par}', g)$ where $tr[\mathsf{m}] = \mathsf{td}$. Thus, it follows from the translatability of TLF, or more concretely from the translation completeness, that the view of $\mathcal{A}$ is not changed. We get

$$\Pr[\mathbf{G}_2 \Rightarrow 1] = \Pr[\mathbf{G}_3 \Rightarrow 1].$$

Game $\mathbf{G}_4$: In this game, we let the game abort if $(\mathsf{par}', g) \notin \mathsf{Reg}$, where Reg is the set from the regularity definition of TLF. By regularity of TLF, we have

$$|\Pr[\mathbf{G}_3 \Rightarrow 1] - \Pr[\mathbf{G}_4 \Rightarrow 1]| \leq \varepsilon_{\mathsf{r}}.$$

Game $\mathbf{G}_5$: In this game, we change the signing oracle again. Specifically, we change the commitment and opening phase. Recall that until now, in the commitment phase for an honest party i in a signer set $S \subseteq [n]$ and message m, an element $r_i \xleftarrow{\$} \mathcal{D}$ is sampled and the party sends a commitment $\mathsf{com}_i := \hat{\mathsf{H}}(S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$ for $R_i^{(1)} := \mathsf{T}(g, r_i)$, $R_i^{(2)} := \mathsf{T}(h, r_i)$, and $\mathsf{pk}_i^{(2)} := \mathsf{Translate}(tr[\mathsf{m}], \mathsf{pk}_i)$. As before, h is defined as $h := \mathsf{H}(\mathsf{m})$. Later, in the opening phase, the party sends $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$. Now, we change this as follows: The signing oracle computes $\mathsf{pk}_i^{(2)}$ as in $\mathbf{G}_4$, but it does not compute $R_i^{(1)}, R_i^{(2)}$ and instead sends a random commitment $\mathsf{com}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$ on behalf of party i. It also inserts an entry (S, i, com_i) into a list Sim that keeps track of these simulated commitments. If there is already an $(S', i') \neq (S, i)$ such that $(S', i', \mathsf{com}_i) \in \mathsf{Sim}$, then the game aborts. Note that there are two situations where the preimage of com_i has to be revealed. Namely, $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$ has to be given to the adversary in the opening phase, and whenever party i is corrupted the game needs to output r_i. To handle this, consider the opening phase or the case where party i is corrupted before it reaches the opening phase. Here, we let the game sample $r_i \xleftarrow{\$} \mathcal{D}$ and define $R_i^{(1)} := \mathsf{T}(g, r_i)$ and $R_i^{(2)} := \mathsf{T}(h, r_i)$. Then, the game checks if $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] = \bot$. If it is not, the game aborts. Otherwise, it programs $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] := \mathsf{com}_i$ and continues. That is, in the opening phase it would output $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$, and during a corruption, it would output r_i as part of its state. If a corruption occurs after the opening phase, then r_i has already been defined, and corruption is handled as before. Clearly, the view of $\mathcal{A}$ is only affected by this change if $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$ matches a previous query of $\mathcal{A}$ or the same commitment has been sampled by the game twice. The latter event occurs only with probability $Q_S^2/2^{2\lambda}$ by a union bound over all pairs of queries. To bound the former event, we use the regularity of TLF, which implies that $R_i^{(1)}$ is uniform over the range $\mathcal{R}$. Now, for each fixed pair of signing query and random oracle query, the random oracle query matches $R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}$ with probability at most $1/|\mathcal{R}|$. Thus, the event occurs only with probability $Q_S Q_{\hat{\mathsf{H}}}/2^{2\lambda}$. We get

$$|\Pr[\mathbf{G}_4 \Rightarrow 1] - \Pr[\mathbf{G}_5 \Rightarrow 1]| \leq \frac{Q_S Q_{\hat{\mathsf{H}}}}{|\mathcal{R}|} + \frac{Q_S^2}{2^{2\lambda}}.$$

Game $\mathbf{G}_6$: In this game, we rule out collisions for random oracle $\hat{\mathsf{H}}$. Namely, the game aborts if there are $x \neq x'$ such that $\hat{h}[x] = \hat{h}[x'] \neq \bot$. Clearly, we have

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{Q_{\hat{\mathsf{H}}}^2}{2^{2\lambda}}.$$

Subsequent games will internally make use of an algorithm $\hat{\mathsf{H}}^{-1}$. On input y the algorithm searches for an x such that $\hat{h}[x] = y$. If no such x is found, or if multiple x are found, then the algorithm returns $\bot$. Otherwise, it returns x. Note that in the latter case the game would abort anyways, and so we can assume that if there is a preimage of y, then this preimage is uniquely determined by y.

Game $\mathbf{G}_7$: In this game, we introduce a list Pending and associated algorithms UpdatePending and AddToPending to manage this list. The algorithms are presented as pseudocode in Figure A.18. Intuitively, the list keeps track of honest users i and signing sessions sid for which the game can not yet extract preimages of all commitments sent in the commitment phase. More precisely, the list contains a tuple $(sid, i, \mathcal{M}_1)$ if and only if the following two conditions hold:

- The opening phase oracle $\textsc{Sig}_1(sid, i, \mathcal{M}_1)$ has been called with valid inputs, i.e., for this query the game did not output $\bot$ due to $\mathsf{Allowed}(sid, i, 1, \mathcal{M}_1) = 0$, and at that point the following was true: For every commitment com_j in $\mathcal{M}_1$ such that $(S, j, \mathsf{com}_j) \notin \mathsf{Sim}$, we have $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) \neq \bot$ and with $(S', k, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}) := \hat{\mathsf{H}}^{-1}(\mathsf{com}_j)$ we have $S' = S$ and $k = j$, where S is the signer set associated with sid.
- There is a commitment com_j in $\mathcal{M}_1$ such that $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) = \bot$.

To ensure that the list satisfies this invariant, we add a triple $(sid, i, \mathcal{M}_1)$ to Pending when the first condition holds. This is done by algorithm AddToPending. Concretely, whenever $\mathcal{A}$ calls $\textsc{Sig}_1(sid, i, \mathcal{M}_1)$, the oracle returns $\bot$ in case $\mathsf{Allowed}(sid, i, 1, \mathcal{M}_1) = 0$. If $\mathsf{Allowed}(sid, i, 1, \mathcal{M}_1) = 1$, the game immediately calls $\mathsf{AddToPending}(sid, i, 1, \mathcal{M}_1)$, which checks the first condition of the invariant and inserts the tripe $(sid, i, 1, \mathcal{M}_1)$ into Pending if it holds. Then, the game continues the simulation of $\textsc{Sig}_1$ as before. Further, we invoke algorithm UpdatePending whenever the map $\hat{h}$ is changed, i.e., during queries to $\hat{\mathsf{H}}$, and in corruption and signing oracles (see $\mathbf{G}_5$). On every invocation, the algorithm does the following:

1. Initialize an empty list New.
2. Iterate trough all entries $(sid, i, \mathcal{M}_1)$ in Pending, and do the following:
   (a) Check if the entry has to be removed because it is violating the invariant. That is, check if for all j in the signer set S associated with session sid, we have $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) \neq \bot$, where $\mathcal{M}_1 = (\mathsf{com}_j)_{j \in S}$. If this is not the case, skip this entry and keep it in Pending.
   (b) We know that for all indices $j \in S$, the value $(S'_j, k_j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) = \hat{\mathsf{H}}^{-1}(\mathsf{com}_j)$ exists. Further, it must hold that $S'_j = S$ and $k_j = j$, as otherwise this entry would not have been added to Pending in the first place. Remove the entry from Pending, and determine the combined nonces and secondary public key
   $$R^{(1)} = \sum_{j \in S} R_j^{(1)}, \quad R^{(2)} = \sum_{j \in S} R_j^{(2)}, \quad \mathsf{pk}^{(2)} = \sum_{j \in S} \ell_{j,S} \mathsf{pk}_j^{(2)}.$$
   (c) Let m be the message associated with the session sid.
   (d) If $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m}) \notin \mathsf{New}$ but $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \neq \bot$, abort the execution of the entire game (see bad event Defined below).
   (e) Otherwise, sample $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \xleftarrow{\$} \mathcal{S}$ and insert $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})$ into New.

To summarize, this algorithm removes all entries violating the invariant from the list Pending. For each such entry that is removed, the algorithm computes the combined nonces $R^{(1)}, R^{(2)}$ and secondary public key $\mathsf{pk}^{(2)}$. Roughly, it aborts the execution, if random oracle $\bar{\mathsf{H}}$ for these inputs is already defined. List New ensures that the abort is not triggered if the algorithm itself programmed $\bar{h}$ in a previous iteration within the same invocation. In addition to algorithm UpdatePending, we introduce the following events, on which the game aborts its execution:

- Event BadQuery: This event occurs, if for a random oracle query to $\hat{\mathsf{H}}$ for which the hash value is not yet defined and freshly sampled as $\mathsf{com} \xleftarrow{\$} \{0,1\}^{2\lambda}$, there is an entry $(sid, i, \mathcal{M}_1)$ in Pending such that com is in $\mathcal{M}_1$.

- Event Defined: This event occurs, if the execution is aborted during algorithm UpdatePending.

For shorthand notation, we set Bad := BadQuery ∨ Defined. The probability of BadQuery can be bounded as follows: Fix a random oracle query to $\hat{\mathsf{H}}$ for which the hash value is not yet defined. Fix an entry $(sid, i, \mathcal{M}_1)$. Note that over the entire game, there are at most Q_S of these entries. Further, fix an index $j \in [t+1]$. The probability that com collides with the jth entry of $\mathcal{M}_1$ is clearly at most $1/2^{2\lambda}$. With a union bound over all triples of queries, entries, and indices, we get that the probability of BadQuery is at most $Q_{\hat{\mathsf{H}}}Q_S(t+1)/2^{2\lambda}$. Next, we bound the probability of Defined assuming BadQuery does not occur. Under this assumption, one can easily observe that when an entry is removed from list Pending and $R^{(1)} = \sum_{j\in S} R_j^{(1)}$ is the combined first nonce, then there is an $j^* \in S$ such that the game sampled $R_{j^*}^{(1)}$ just before invoking algorithm UpdatePending. Precisely, it must have set $R_{j^*}^{(1)} := \mathsf{T}(g, r)$ for some random $r \xleftarrow{\$} \mathcal{D}$. By regularity of TLF, this means $R_{j^*}^{(1)}$ is uniform over $\mathcal{R}$, and this means that the combined first nonce $R^{(1)}$ is also uniform. Thus for any fixed entry of in Pending, the probability that $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}]$ is already defined when the entry is removed, is at most $Q_{\hat{\mathsf{H}}}/|\mathcal{R}|$. With a union bound over all entries we can now bound the probability of Defined by $Q_{\hat{\mathsf{H}}}Q_S/|\mathcal{R}|$. In combination, we get

$$\Pr[\mathsf{Bad}] \leq \Pr[\mathsf{BadQuery}] + \Pr[\mathsf{Defined} \mid \neg\mathsf{BadQuery}] \leq \frac{Q_{\hat{\mathsf{H}}}Q_S(t+1)}{2^{2\lambda}} + \frac{Q_{\hat{\mathsf{H}}}Q_S}{|\mathcal{R}|}.$$

and thus

$$|\Pr[\mathbf{G}_6 \Rightarrow 1] - \Pr[\mathbf{G}_7 \Rightarrow 1]| \leq \Pr[\mathsf{Bad}] \leq \frac{Q_{\hat{\mathsf{H}}}Q_S(t+1)}{2^{2\lambda}} + \frac{Q_{\hat{\mathsf{H}}}Q_S}{|\mathcal{R}|}.$$

Game $\mathbf{G}_8$: In this game, we change algorithm UpdatePending. Specifically, we change what we insert into list New. Recall from the previous game that when we removed an entry $(sid, i, \mathcal{M}_1)$ from Pending, we aborted the game if $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m}) \notin \mathsf{New}$ but $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \neq \bot$. Otherwise, we inserted tuples $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})$. Now, we instead abort if $(S, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m}) \notin \mathsf{New}$ but $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \neq \bot$, and otherwise insert $(S, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})$, where S is the signer set associated with session sid. One can see that the two games can only differ if for two entries $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}_1')$ that are removed from Pending in the same invocation of UpdatePending, the signer sets S and S' differ but the respective tuples $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})$ and $(R'^{(1)}, R'^{(2)}, \mathsf{pk}'^{(2)}, \mathsf{m}')$ are the same and $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \neq \bot$. In this case, game $\mathbf{G}_8$ would abort, but game $\mathbf{G}_7$ would not. We argue that this can not happen: Assume that two entries $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}_1')$ with associated signer sets S and S' are removed from Pending. Then, we know that algorithm UpdatePending has been invoked because the game programmed $\hat{h}$ at some point, say $\hat{h}[S_*, j_*, R_*^{(1)}, R_*^{(2)}, \mathsf{pk}_*^{(2)}] := \mathsf{com}_*$, such that com_* is in both $\mathcal{M}_1$ and $\mathcal{M}_1'$. Thus, the algorithm only removes the entry $(sid, i, \mathcal{M}_1)$ from the list if the first component of $\hat{\mathsf{H}}^{-1}(\mathsf{com}_*)$ is S, i.e., if $S_* = S$. Similarly, it only removes the entry $(sid', i', \mathcal{M}_1')$ if the first the first component of $\hat{\mathsf{H}}^{-1}(\mathsf{com}_*)$ is S', i.e., if $S_* = S'$. Thus, it only removes both if $S = S_* = S'$. With that, we have

$$\Pr[\mathbf{G}_7 \Rightarrow 1] = \Pr[\mathbf{G}_8 \Rightarrow 1].$$

Game $\mathbf{G}_9$: We introduce two more algorithms, presented as pseudocode in Figure A.18. Intuitively, these allow us to group tuples of the form $(sid, i, \mathcal{M}_1)$ that have been inserted into list Pending into equivalence classes. To be clear, the relation is defined on all triples in Pending and on all triples that already have been removed from Pending, but not on any other entries. The intuition, roughly, is that such triples lead to the same combined nonces if and only if they are in the same equivalence class. The effect of this is will be that we know the challenge just from the tuple $(sid, i, \mathcal{M}_1)$. We now turn to the details. We introduce an algorithm Equivalent that takes as input two triples $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}_1')$ and decides whether they are equivalent as follows:

1. Let S, S' and m, m' be the signer sets and messages associated with sessions sid and sid', respectively. If $S \neq S'$ or $\mathsf{m} \neq \mathsf{m}'$, the triples are not equivalent.
2. Thus, assume $S = S'$ and write $\mathcal{M}_1 = (\mathsf{com}_j)_{j \in S}$ and $\mathcal{M}'_1 = (\mathsf{com}'_j)_{j \in S}$. Let $F \subseteq S$ (resp. $F' \subseteq S'$) be the set of indices $j \in S$ (resp $j \in S'$) such that $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) = \bot$ (resp. $\hat{\mathsf{H}}^{-1}(\mathsf{com}'_j) = \bot$). If $(\mathsf{com}_j)_{j \in F} \neq (\mathsf{com}'_j)_{j \in F'}$, then the triples are not equivalent.
3. Define $\bar{F} := S \setminus F$ and $\bar{F}' := S \setminus F'$. For each $j \in \bar{F}$, we know that the value $(\tilde{S}_j, k_j, {R'_j}^{(1)}, {R'_j}^{(2)}, {\mathsf{pk}'_j}^{(2)}) = \hat{\mathsf{H}}^{-1}(\mathsf{com}'_j)$ exists. Similarly, for each $j \in \bar{F}'$, we know that the value $(\tilde{S}'_j, k'_j, {R'_j}^{(1)}, {R'_j}^{(2)}, {\mathsf{pk}'_j}^{(2)}) = \hat{\mathsf{H}}^{-1}(\mathsf{com}'_j)$ exists. With these, we can define partially combined nonces and secondary keys

$$\begin{array}{lll} \bar{R}^{(1)} := \sum_{j \in \bar{F}} R_j^{(1)}, & \bar{R}^{(2)} := \sum_{j \in \bar{F}} R_j^{(2)} & \bar{\mathsf{pk}}^{(2)} := \sum_{j \in \bar{F}} \ell_{j,S} \mathsf{pk}_j^{(2)} \\ \bar{R}'^{(1)} := \sum_{j \in \bar{F}'} {R_j}'^{(1)}, & \bar{R}'^{(2)} := \sum_{j \in \bar{F}'} {R_j}'^{(2)} & \bar{\mathsf{pk}}'^{(2)} := \sum_{j \in \bar{F}'} \ell_{j,S} {\mathsf{pk}_j}'^{(2)}. \end{array}$$

   The triples are not equivalent, if $(\bar{R}^{(1)}, \bar{R}^{(2)}, \bar{\mathsf{pk}}^{(2)}) \neq (\bar{R}'^{(1)}, \bar{R}'^{(2)}, \bar{\mathsf{pk}}'^{(2)})$. Otherwise, they are equivalent.

In summary, two triples are equivalent if their signer sets, messages, partially combined nonces and secondary public keys, and remaining commitments match. It is clear that at any fixed point in time during the experiment, this is indeed an equivalence relation. In the following two claims, we argue that this relation is preserved over time. For that, we first make some preliminary observations, using notation as in the definition of equivalence above:

1. The equivalence relation can potentially only change when oracle $\hat{\mathsf{H}}$ is updated during queries to $\textsc{Sig}_1$ (i.e., the opening phase) or during corruption queries, which may make the sets F and F' change. This is because triples are only inserted into $\mathsf{Pending}$ if the only commitments without preimages are simulated, and the preimages of these are only set in such calls (see $\mathbf{G}_7$).
2. The sets F and F' can only get smaller over time, as we assume that no collisions occur.
3. When the oracle is programmed during such calls, say by setting $\hat{h}[S_*, j_*, R_*^{(1)}, R_*^{(2)}, \mathsf{pk}_*^{(2)}] := \mathsf{com}_*$, then it must hold that $(S_*, j_*, \mathsf{com}_*) \in \mathsf{Sim}$. In particular, if in this case some j is removed from F (or F') because com_j (or com'_j) now has a preimage, then it must hold that $\mathsf{com}_* = \mathsf{com}_j$ and $j_* = j$. This is because otherwise, if $j \neq j_*$, then we would have $(\tilde{S}, j, \mathsf{com}_*) \in \mathsf{Sim}$ for some $\tilde{S}$ (because the entry was added to $\mathsf{Pending}$) and $(S_*, j_*, \mathsf{com}_*) \in \mathsf{Sim}$, and such a collision was ruled out in $\mathbf{G}_5$.
4. Again, assume that the oracle is programmed during such calls as $\hat{h}[S_*, j_*, R_*^{(1)}, R_*^{(2)}, \mathsf{pk}_*^{(2)}] := \mathsf{com}_*$. Now, assume that both F and F' change. Then, we know (because of the previous observation), that the same $j = j_*$ is removed from both F and F', and $\mathsf{com}_j = \mathsf{com}_* = \mathsf{com}'_j$ is removed from both $(\mathsf{com}_j)_{j \in F}$ and $(\mathsf{com}'_j)_{j \in F'}$. Thus, these lists are the same before the update if and only if they are the same after the update.
5. In the setting of the previous observation, denote the point in time before the update as t_0, and the point in time after the update as t_1. Further, denote the associated partially combined nonces and secondary public keys at time t_b for $b \in \{0, 1\}$ by

$$\bar{R}_b^{(1)},\ \bar{R}_b^{(2)},\ \bar{\mathsf{pk}}_b^{(2)},\ \text{and}\ \bar{R}_b'^{(1)},\ \bar{R}_b'^{(2)},\ \bar{\mathsf{pk}}_b'^{(2)}.$$

   Now, we observe that

$$\bar{R}_1^{(1)} = \bar{R}_0^{(1)} + R_*^{(1)},\ \ \bar{R}_1^{(2)} = \bar{R}_0^{(2)} + R_*^{(2)},\ \ \bar{\mathsf{pk}}_1^{(2)} = \bar{\mathsf{pk}}_0^{(2)} + \ell_{j_*,S_*} \mathsf{pk}_*^{(2)}.$$

The same holds for $\bar{R}_b^{\prime(1)}$, $\bar{R}_b^{\prime(2)}$, and $\bar{\mathsf{pk}}_b^{\prime(2)}$. Therefore, we see that

$$(\bar{R}_0^{(1)}, \bar{R}_0^{(2)}, \bar{\mathsf{pk}}_0^{(2)}) = (\bar{R}_0^{\prime(1)}, \bar{R}_0^{\prime(2)}, \bar{\mathsf{pk}}_0^{\prime(2)})$$
$$\text{if and only if } (\bar{R}_1^{(1)}, \bar{R}_1^{(2)}, \bar{\mathsf{pk}}_1^{(2)}) = (\bar{R}_1^{\prime(1)}, \bar{R}_1^{\prime(2)}, \bar{\mathsf{pk}}_1^{\prime(2)}).$$

Now, we show that the equivalence relation does not change over time, using our notation from above and the observations we made.

Equivalence Claim 1. If two triples $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}_1')$ are equivalent at some point in time, then they stay equivalent for the rest of the game.

Proof of Equivalence Claim 1. Both signer set and message do not change over time. For the other components that determine whether the triples are equivalent, we consider two cases: Either, on an update of $\hat{\mathsf{H}}$, both do not change. In this case the triples trivially stay equivalent. In the other case, both of them change, as the lists $(\mathsf{com}_j)_{j\in F}$ and $(\mathsf{com}'_j)_{j\in F'}$ are the same before the update. Now, it easily follows from our last observation above that the triples stay equivalent.

Equivalence Claim 2. If two triples $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}_1')$ are not equivalent at some point in time, then the probability that they become equivalent later is negligible. Concretely, if Converge is the event that any two non-equivalent triples become equivalent at some point in time, then

$$\Pr[\mathsf{Converge}] \leq \frac{Q_S^2(Q_S + t)}{|\mathcal{R}|}.$$

Proof of Equivalence Claim 2. Clearly, if $\mathsf{m} \neq \mathsf{m}'$ or $S \neq S'$, then the triples will stay non-equivalent. Now, consider an update of $\hat{\mathsf{H}}$ that is caused by a query to $\textsc{Sig}_1$ or the corruption oracle and will potentially change the equivalence relation. We consider two cases: In the first case, the lists $(\mathsf{com}_j)_{j\in F}$ and $(\mathsf{com}'_j)_{j\in F'}$ are the same before the update. In this case, they either do not change, in which case the triples trivially stay non-equivalent, or they both change, in which case it follows from our last observation above that they stay non-equivalent. In the second case, the lists $(\mathsf{com}_j)_{j\in F}$ and $(\mathsf{com}'_j)_{j\in F'}$ are different before the update. If they stay different after the update, the triples stay non-equivalent. If they become the same after the update, this means that an entry was removed from only one of them, say $j = j_*$ from F and thus $\mathsf{com}_j = \mathsf{com}_*$ from $(\mathsf{com}_j)_{j\in F}$. For this case, use notation $\bar{R}_b^{(1)}$ and $\bar{R}_b^{\prime(1)}$ as in the last last observation above and notice that $\bar{R}_1^{\prime(1)} = \bar{R}_0^{\prime(1)}$ because $(\mathsf{com}'_j)_{j\in F'}$ is not changed during the update. On the other hand, $(\mathsf{com}_j)_{j\in F}$ is changed by the update and we have $\bar{R}_1^{(1)} = \bar{R}_0^{(1)} + R_*^{(1)}$. Thus, if the triples become equivalent, we must have

$$\bar{R}_0^{\prime(1)} = \bar{R}_1^{\prime(1)} = \bar{R}_1^{(1)} = \bar{R}_0^{(1)} + R_*^{(1)}.$$

Notice that $R_*^{(1)}$ is sampled in the signing or corruption oracle by sampling some $r_* \xleftarrow{\$} \mathcal{D}$ and setting $R_*^{(1)} = \mathsf{T}(g, r_*)$. Thus, $R_*^{(1)}$ is uniformly distributed over $\mathcal{R}$ by the regularity of TLF and independent of $\bar{R}_0^{\prime(1)}$ and $\bar{R}_0^{(1)}$, which means that this equation holds with probability at most $1/|\mathcal{R}|$. Taking a union bound over all pairs of triples and all queries to the signing oracle and the corruption oracle, the claim follows.

With our equivalence relation at hand, we introduce an algorithm GetChallenge that behaves as a random oracle on equivalence classes. That is, it assigns each class a random challenge $c \xleftarrow{\$} \mathcal{S}$ in a lazy manner. More precisely, it gets as input a triple $(sid, i, \mathcal{M}_1)$ and checks if a triple in the same equivalence class[5] is already assigned a challenge c. This is done using algorithm Equivalent. If so, it returns this challenge c. If not, it assigns a random challenge $c \xleftarrow{\$} \mathcal{S}$ to the triple $(sid, i, \mathcal{M}_1)$.

[5] It is essential for this algorithm that we have shown that equivalence classes are preserved over time. Otherwise, the behavior of this algorithm would be ambiguous.

These two new algorithms are used in the following way: Recall that in previous games, algorithm UpdatePending would program $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \xleftarrow{\$} \mathcal{S}$ whenever an entry $(sid, i, \mathcal{M}_1)$ is removed from Pending and no abort occurs, where $\mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}$ are the corresponding secondary public keys, combined nonces, and messages. Now, instead of sampling $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}]$ at random, the algorithm sets $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] := \mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$. We need to argue that this way of programming the random oracle does not change the view of the adversary. Concretely, all we need to argue is that two different inputs $x \neq x'$ to random oracle $\bar{\mathsf{H}}$ get independently sampled outputs. Clearly, it is sufficient to consider inputs of the form

$$x = (\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}), \quad x' = (\mathsf{pk}, \mathsf{pk}^{'(2)}, R^{'(1)}, R^{'(2)}, \mathsf{m}'),$$

which both are covered by the newly introduced programming in algorithm UpdatePending. Let $(sid, i, \mathcal{M}_1)$ be the entry removed from Pending associated with x and $(sid', i', \mathcal{M}'_1)$ be the entry removed from Pending associated with x'. Consider the point in time where the second entry, say $(sid', i', \mathcal{M}'_1)$ has been removed. One can see that the outputs $\bar{\mathsf{H}}(x)$ and $\bar{\mathsf{H}}(x')$ are independent, unless at this point in time $(sid, i, \mathcal{M}_1)$ and $(sid', i', \mathcal{M}'_1)$ were equivalent. However, by definition of equivalence (algorithm Equivalent), them being equivalent would mean that $\mathsf{m} = \mathsf{m}'$ and $(\mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}) = (\mathsf{pk}^{'(2)}, R^{'(1)}, R^{'(2)})$, as the sets F and F' are both empty because both entries have been removed from Pending. Thus, we would have $x = x'$. This shows that the distribution of random oracle outputs does not change, and so we have

$$\Pr[\mathbf{G}_8 \Rightarrow 1] = \Pr[\mathbf{G}_9 \Rightarrow 1].$$

Game $\mathbf{G}_{10}$: In this game, we change the signing oracle and corruption oracle. Roughly, we use an honest-verifier zero-knowledge-style simulation to simulate signing without secret keys. Intuitively, we can do that, because now we know the challenge already in the opening phase before fixing nonces. More precisely, recall that until now, signers in the opening phase, i.e., on a query $\textsc{Sig}_1(sid, i, \mathcal{M}_1)$, sampled a random $r_i \xleftarrow{\$} \mathcal{D}$ and set $R_i^{(1)} := \mathsf{T}(g, r_i)$ and $R_i^{(2)} := \mathsf{T}(h, r_i)$. Later, in the response phase, the signer sent $s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$ where $c := \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$ and $\mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}$ are the combined secondary public key and nonces. Additionally, when the signer is corrupted, it has to send r_i as part of its state. We change this as follows: In the opening phase, consider two cases: First, if $(sid, i, \mathcal{M}_1)$ has not been added to the list Pending, then the signer sets $c := 0$. Observe that in this case, we can assume that the signer never reaches the response phase for this session due to our changes in $\mathbf{G}_6$ and $\mathbf{G}_7$. Otherwise, it sets $\tilde{c} := \mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$. In both cases, the signer samples $s_i \xleftarrow{\$} \mathcal{D}$ and sets $R_i^{(1)} := \mathsf{T}(g, s_i) - \tilde{c} \cdot \ell_{i,S} \cdot \mathsf{pk}_i$ and $R_i^{(2)} := \mathsf{T}(h, s_i) - \tilde{c} \cdot \ell_{i,S} \cdot \mathsf{pk}_i^{(2)}$. Later, when the signer has to output something in the response phase, it outputs the s_i that it sampled in the opening phase. Further, when the signer is corrupted after the opening phase, it sets $r_i := s_i - \tilde{c} \cdot \ell_{i,S} \cdot \mathsf{sk}_i$. To argue indistinguishability, we need to show that $\tilde{c}$ and $c = \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$ are the same. This is established as follows:

1. When the signer is queried in the response phase and does not return $\perp$, we know that the entry $(sid, i, \mathcal{M}_1)$ has been removed from Pending.
2. When it was removed from the list, the combined nonce and secondary public key that have been computed are exactly $R^{(1)}, R^{(2)}$, and $\mathsf{pk}^{(2)}$.
3. Therefore, in the invocation of UpdatePending in which the entry was removed from the list, one of two events happened:
   (a) Either $\bar{h}$ has been programmed as $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] := \mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$;
   (b) Or, $\bar{h}$ has been programmed as $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] := \mathsf{GetChallenge}(sid', i', \mathcal{M}'_1)$ for some triple $(sid', i', \mathcal{M}'_1)$ with the same associated signer set S (see $\mathbf{G}_8$) and message

m. In this case, we know that $(sid', i', \mathcal{M}_1')$ is equivalent to $(sid, i, \mathcal{M}_1)$ and therefore $\mathsf{GetChallenge}(sid', i', \mathcal{M}_1')$ returned the same as what the query $\mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$ would have returned at that point.

4. Thus, we only need to argue that the output of $\mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$ did not change over time. This follows from our claims about the stability of equivalence classes over time, assuming event Converge does not occur.

We get

$$|\Pr[\mathbf{G}_9 \Rightarrow 1] - \Pr[\mathbf{G}_{10} \Rightarrow 1]| \leq \Pr[\mathsf{Converge}] \leq \frac{Q_S^2(Q_S + t)}{|\mathcal{R}|}.$$

Game $\mathbf{G}_{11}$: We change the game by no longer assuming that $(\mathsf{par}', g) \in \mathsf{Reg}$. Clearly, we have

$$|\Pr[\mathbf{G}_{10} \Rightarrow 1] - \Pr[\mathbf{G}_{11} \Rightarrow 1]| \leq \varepsilon_{\mathsf{r}}.$$

It remains to bound the probability that game $\mathbf{G}_{11}$ outputs 1. Before turning to that, we emphasize the main property we have established via our changes: We do not longer need secret key shares sk_i to simulate the signer oracle. We only need them on corruption queries. Now, we bound the probability that game $\mathbf{G}_{11}$ outputs 1 by considering two events, depending on the final forgery (m^*, σ^*) with $\sigma^* = (\mathsf{pk}^{*(2)}, c^*, s^*)$:

- Event Orthogonal: This event occurs, if $\mathbf{G}_{11}$ outputs 1, and there is no $x_0 \in \mathcal{D}$ such that $\mathsf{T}(g, x_0) = \mathsf{pk}$ and $\mathsf{T}(h^*, x_0) = \mathsf{pk}^{*(2)}$, where $h^* = \mathsf{H}(\mathsf{m}^*)$.
- Event Parallel: This event occurs, if $\mathbf{G}_{11}$ outputs 1, and there is a $x_0 \in \mathcal{D}$ such that $\mathsf{T}(g, x_0) = \mathsf{pk}$ and $\mathsf{T}(h^*, x_0) = \mathsf{pk}^{*(2)}$, where $h^* = \mathsf{H}(\mathsf{m}^*)$.

Clearly, we have

$$\Pr[\mathbf{G}_{11} \Rightarrow 1] \leq \Pr[\mathsf{Orthogonal}] + \Pr[\mathsf{Parallel}].$$

We bound the probability of these events separately. For event Orthogonal, we make use of Lemma 5.1. Concretely, we sketch a reduction that runs in the game specified in Lemma 5.1, such that the event bounded in Lemma 5.1 occurs if event Orthogonal occurs and some guesses of the reduction were correct. Namely, the reduction gets as input parameters par' for TLF. It samples $i^* \xleftarrow{\$} [Q_{\bar{\mathsf{H}}}]$ and simulates $\mathbf{G}_{11}$ for $\mathcal{A}$ except for the i^*th random oracle query to $\bar{\mathsf{H}}$. Let that query be $\bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$. The reduction aborts if the hash value is already defined. Otherwise, the reduction outputs its state, $g, h := \mathsf{H}(\mathsf{m}), X_1 := \mathsf{pk}, X_2 := \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}$ to the game from Lemma 5.1, receiving a challenge $c \in \mathcal{S}$ in return. It programs $\bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}) := c$ and continues the simulation. Later, when $\mathcal{A}$ outputs its forgery, the reduction aborts if the query defining c^* was not the i^*th query to $\bar{\mathsf{H}}$. Otherwise, it outputs s^* to the game from Lemma 5.1. Note that one of the random oracle queries to $\bar{\mathsf{H}}$ (possibly the one made by the game during verification) has to be the one defining c^*. Especially, the random oracle is not reprogrammed at that position because $\mathcal{A}$ is not allowed to make a signing query for m^*. Also, it is clear that if Orthogonal occurs and the guess was correct, then the event in Lemma 5.1 occurs. As the view of $\mathcal{A}$ is independent of i^* until it terminates, we have

$$\Pr[\mathsf{Orthogonal}] \leq \frac{Q_{\bar{\mathsf{H}}}}{|\mathcal{S}|}.$$

Next, we bound the probability of event Parallel. For that, we describe a reduction $\mathcal{B}$ against the t-algebraic translation resistance of TLF:

1. $\mathcal{B}$ gets as input parameters par' for TLF, tags g, h, and images $X_0, \ldots, X_t \in \mathcal{R}$.
2. $\mathcal{B}$ simulates game $\mathbf{G}_{11}$ for $\mathcal{A}$, with the following changes

- $\mathcal{B}$ sets up the key in a different way: Namely, it sets $\mathsf{pk} := \mathsf{pk}_0 := X_0$, and it sets $\mathsf{pk}_i := X_i$ for each $i \in [t]$. Further, let $S_0 := \{0\} \cup [t]$. $\mathcal{B}$ sets $\mathsf{pk}_i := \sum_{j \in S_0} \ell_{j,S_0}(i)\mathsf{pk}_j$ for each $i \in [n] \setminus S_0$. This means that $\mathcal{B}$ is not aware of the associated secret keys sk_i. Note that the public keys are distributed exactly as in $\mathbf{G}_{11}$.
- $\mathcal{B}$ provides all random oracles as in $\mathbf{G}_{11}$, except for random oracle H. Namely, for this oracle, instead of sampling $h[\mathsf{m}]$ at random when $b[\mathsf{m}] = 1$, $\mathcal{B}$ samples $(h', \mathsf{td}) \leftarrow \mathsf{Shift}(\mathsf{par}', h)$ and sets $h[\mathsf{m}] := h'$ and $tr[\mathsf{m}] := \mathsf{td}$.
- $\mathcal{B}$ provides the signing oracles as in $\mathbf{G}_{11}$. Observe that $\mathcal{B}$ does not need any secret keys for that.
- Whenever $\mathcal{A}$ queries the corruption oracle to corrupt user i, the reduction $\mathcal{B}$ queries $x_i := \textsc{Inv}(\ell_{0,S_0}(i), \ldots, \ell_{t,S_0}(i))$. Then, it sets $\mathsf{sk}_i := x_i$ and continues the simulation of the corruption as in $\mathbf{G}_{11}$. It is clear that sk_i is correctly distributed. Further, $\mathcal{B}$ queries its oracle exactly t times because $\mathcal{A}$ corrupts exactly t parties (see $\mathbf{G}_0$).

3. Finally, when $\mathcal{A}$ outputs its forgery (m^*, σ^*) with $\sigma^* = (\mathsf{pk}^{*(2)}, c^*, s^*)$ and $\mathbf{G}_{11}$ outputs 1 (which can be efficiently checked by $\mathcal{B}$), the reduction $\mathcal{B}$ retrieves $\mathsf{td}^* := tr'[\mathsf{m}^*]$. This entry exists because of the change in $\mathbf{G}_1$. Then, it computes $X'_0 := \mathsf{InvTranslate}(\mathsf{td}^*, \mathsf{pk}^{*(2)})$ for $h^* := \mathsf{H}(\mathsf{m}^*)$.

4. To recall, $\mathsf{Corrupted} \subseteq [n]$ is the set of parties $i \in [n]$ such that $\mathcal{A}$ corrupted party i. By our assumption from $\mathbf{G}_0$, we know that $|\mathsf{Corrupted}| = t$. The reduction sets $S^* := \mathsf{Corrupted} \cup \{0\}$, which has size $t + 1$. Further, it sets $X'_i := \mathsf{T}(h, x_i)$ for each $i \in \mathsf{Corrupted}$ and $X'_i := \sum_{j \in S^*} \ell_{j,S^*}(i) X'_j$ for each $i \in [n] \setminus \mathsf{Corrupted}$. It outputs $X'_0, \ldots, X'_t$ to the algebraic translation resistance game.

Clearly, the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. Further the only difference between the view of $\mathcal{A}$ in game $\mathbf{G}_{11}$ and in the simulation provided by $\mathcal{B}$ is the distribution of random oracle H. It follows from the ε_{t}-translatability of TLF and a union bound over all queries that this changes the probability of event $\mathsf{Parallel}$ by at most $Q_{\mathsf{H}}\varepsilon_{\mathsf{t}}$. Now, it remains to argue that if event $\mathsf{Parallel}$ occurs, then $\mathcal{B}$ breaks algebraic translation resistance. So, assume that $\mathsf{Parallel}$ occurs. We have to argue that for each $i \in \{0\} \cup [t]$, there is a z_i such that $\mathsf{T}(g, z_i) = X_i$ and $\mathsf{T}(h, z_i) = X'_i$. First, for all $i \in \mathsf{Corrupted}$ this holds by construction. For $i = 0$, we know (because $\mathsf{Parallel}$) that there is an x_0 such that $\mathsf{T}(g, x_0) = \mathsf{pk}$ and $\mathsf{T}(h^*, x_0) = \mathsf{pk}^{*(2)}$ for $h^* = \mathsf{H}(\mathsf{m}^*)$. Now, we know that $X'_0 = \mathsf{InvTranslate}(\mathsf{td}^*, \mathsf{pk}^{*(2)}) = \mathsf{T}(h, x_0)$, by translation completeness. Thus, it remains to show that the property holds for all $i \in [t]$ with $i \notin \mathsf{Corrupted}$. For these, we have

$$X'_i = \sum_{j \in S^*} \ell_{j,S^*}(i) X'_j = \sum_{j \in S^*} \ell_{j,S^*}(i) \mathsf{T}(h, x_i) = \mathsf{T}\left(h, \sum_{j \in S^*} \ell_{j,S^*}(i) x_i\right).$$

Further, we have

$$\mathsf{T}\left(g, \sum_{j \in S^*} \ell_{j,S^*}(i) x_i\right) = \sum_{j \in S^*} \ell_{j,S^*}(i) \mathsf{T}(g, x_i) = \sum_{j \in S^*} \ell_{j,S^*}(i) \mathsf{pk}'_i = X'_i,$$

by construction. With this, we showed that

$$\Pr[\mathsf{Parallel}] \leq Q_{\mathsf{H}}\varepsilon_{\mathsf{t}} + \mathsf{Adv}^{t\text{-A-TRAN-RES}}_{\mathcal{B},\mathsf{TLF}}(\lambda).$$

This finishes the proof. □

5.5 Instantiations

We instantiate our threshold signature scheme by providing concrete tagged linear function families. Our first instantiation is based on a one-more variant of the CDH assumption, and our second instantiation is based on DDH. The advantage of the latter instantiation is avoiding an interactive assumption, while the former is slightly more efficient.

5.5.1 Instantiation from (Algebraic) One-More CDH

We can instantiate the tagged linear function family by mapping a tag $h \in \mathbb{G}$ and a domain element $x \in \mathbb{Z}_p$ to $h^x \in \mathbb{G}$. Regularity and translatability are easy to show, and algebraic translation resistance follows from an algebraic one-more variant of CDH, namely, from AOMCDH as defined in Section 2.3. More formally, to define the tagged linear function family $\mathsf{TLF}_{\mathsf{AOMCDH}} = (\mathsf{Gen}_{\mathsf{AOMCDH}}, \mathsf{T}_{\mathsf{AOMCDH}})$ based on the AOMCDH assumption, we assume a group generation algorithm GGen. To recall, such an algorithm takes as input 1^λ and outputs the description of a group $\mathbb{G}$ of prime order p with generator $g \in \mathbb{G}$. Algorithm $\mathsf{Gen}_{\mathsf{AOMCDH}}$ runs GGen and outputs parameters $\mathsf{par} = (\mathbb{G}, g, p)$. These parameters define the following sets of scalars, tags, and the domain and range, respectively:

$$\mathcal{S} := \mathbb{Z}_p, \;\; \mathcal{T} := \mathbb{G}, \;\; \mathcal{D} := \mathbb{Z}_p, \;\; \mathcal{R} := \mathbb{G}.$$

Clearly, $\mathcal{D}$ and $\mathcal{R}$ are vector spaces over the field $\mathcal{S}$. Given a tag[6] $u \in \mathbb{G}$ and an input $x \in \mathbb{Z}_p$, the tagged linear function $\mathsf{T}_{\mathsf{AOMCDH}}$ is defined as follows

$$\mathsf{T}_{\mathsf{AOMCDH}}(u, x) := u^x \in \mathbb{G}.$$

Clearly, $\mathsf{T}_{\mathsf{AOMCDH}}$ can be computed efficiently and is a homomorphism. It remains to argue regularity, translatability and algebraic translation resistance.

Lemma 5.2. *The tagged linear function family* $\mathsf{TLF}_{\mathsf{AOMCDH}}$ *is* ε_{r}*-regular, where* $\varepsilon_{\mathsf{r}} \leq 1/p$.

Proof. We define the set Reg from the regularity definition to be the set of group parameters and tags $u \in \mathbb{G}$ such that u is a generator of $\mathbb{G}$. Clearly, for parameters $\mathsf{par} \leftarrow \mathsf{Gen}_{\mathsf{AOMCDH}}(1^\lambda)$ and a random tag $u \xleftarrow{\$} \mathbb{G}$, the probability that $(\mathsf{par}, u) \notin \mathsf{Reg}$, i.e., that u is not a generator, is $1/p$. Further, it is clear that if u is a generator, then $\mathsf{T}_{\mathsf{AOMCDH}}(u, \cdot)$ is a bijection from $\mathbb{Z}_p$ to $\mathbb{G}$, and therefore images of uniformly random elements are uniformly random. □

Lemma 5.3. *The tagged linear function family* $\mathsf{TLF}_{\mathsf{AOMCDH}}$ *is* ε_{t}*-translatable, where* $\varepsilon_{\mathsf{t}} \leq 2/p$.

Proof. Algorithm Shift takes as input parameters par and a tag $u \in \mathbb{G}$. It samples $r \xleftarrow{\$} \mathbb{Z}_p^*$ and outputs a tag $h := u^r$ and a trapdoor $\mathsf{td} := r$. Algorithm Translate takes as input the trapdoor $\mathsf{td} = r$ and an element $X \in \mathbb{G}$. It outputs X^r. Algorithm InvTranslate get the same input and outputs $X^{1/r}$. With that, it is clear that the distributions $\mathcal{X}_0$ and $\mathcal{X}_1$ from the definition of translatability are the same conditioning on u and h being generators. Thus, the statistical distance between $\mathcal{X}_0$ and $\mathcal{X}_1$ is at most $2/p$. Further, we have

$$\mathsf{Translate}(\mathsf{td}, \mathsf{T}_{\mathsf{AOMCDH}}(u, x)) = (u^x)^r = (u^r)^x = \mathsf{T}_{\mathsf{AOMCDH}}(h, x)$$

for $(h, \mathsf{td} = r) \in \mathsf{Shift}(\mathsf{par}, u)$. The inverse direction follows similarly. □

Lemma 5.4. *Let* $t \in \mathbb{N}$ *be a number polynomial in* λ*. If the* AOMCDH *assumption holds relative to* GGen*, then* $\mathsf{TLF}_{\mathsf{AOMCDH}}$ *is* t*-algebraic translation resistant. Concretely, for any PPT algorithm* $\mathcal{A}$ *there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ *and*

$$\mathsf{Adv}^{t\text{-A-TRAN-RES}}_{\mathcal{A},\mathsf{TLF}_{\mathsf{AOMCDH}}}(\lambda) \leq \mathsf{Adv}^{\mathsf{AOMCDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{1}{p}.$$

[6] In this section, we do not use the symbol g for arbitrary tags, as it is reserved for the fixed group generator.

Proof. The proof is trivial, as the game for the AOMCDH assumption is almost exactly the game for t-algebraic translation resistance of $\mathsf{TLF}_{\mathsf{AOMCDH}}$. Note that the only difference is that in the t-algebraic translation resistance game for $\mathsf{TLF}_{\mathsf{AOMCDH}}$, the inputs have the form $X_i = u^{x_i}$ for a random $u \xleftarrow{\$} \mathbb{G}$, whereas in the AOMCDH game, the inputs have the form g^{x_i}, where g is the fixed generator of $\mathbb{G}$. We can easily build a reduction that resolves this discrepancy: Namely, the reduction gets as input the values g^{x_i} from the AOMCDH assumption. It samples $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and sets $u := g^\alpha$ and defines the inputs X_i as $X_i = (g^{x_i})^\alpha = u^{x_i}$. Oracle queries to Inv and the final output of $\mathcal{A}$ are simply forwarded by the reduction. Clearly, the reduction perfectly simulates the t-algebraic translation resistance game. Also, assuming $\alpha \neq 0$ and that $\mathcal{A}$ breaks t-algebraic translation resistance, we see that the reduction breaks AOMCDH. □

5.5.2 Instantiation from DDH

Here, we present our construction $\mathsf{TLF}_{\mathsf{DDH}} = (\mathsf{Gen}_{\mathsf{DDH}}, \mathsf{T}_{\mathsf{DDH}})$ of a tagged linear function family based on the DDH assumption. For that, let GGen be a group generation algorithm that takes as input 1^λ and outputs the description of a group $\mathbb{G}$ of prime order p, along with some generator $g \in \mathbb{G}$. Recall that the DDH assumption (formally defined in Section 2.3) states that it is hard to distinguish tuples $(\mathbb{G}, p, g, h, g^a, h^a)$ from tuples $(\mathbb{G}, p, g, h, u, v)$, where $\mathbb{G}$ is a cyclic group with generator g and prime order p, h, u, v are random group elements, and $a \in \mathbb{Z}_p$ is a random exponent. Algorithm $\mathsf{Gen}_{\mathsf{DDH}}$ simply runs GGen and outputs the description of $\mathbb{G}$, p, and g as parameters par. We make use of the implicit notation for group elements from [EHK+13]. That is, we write $[\mathbf{A}] \in \mathbb{G}^{r\times l}$ for the matrix of group elements with exponents given by the matrix $\mathbf{A} \in \mathbb{Z}_p^{r\times l}$. Precisely, if $\mathbf{A} = (A_{i,j})_{i\in[r],j\in[l]}$, then $[\mathbf{A}] := (g^{A_{i,j}})_{i\in[r],j\in[l]}$. With this notation, observe that one can efficiently compute $[\mathbf{AB}]$ for any matrices $\mathbf{A} \in \mathbb{Z}_p^{r\times l}$, $\mathbf{B} \in \mathbb{Z}_p^{l\times s}$ with matching dimensions from either $[\mathbf{A}]$ and $\mathbf{B}$ or from $\mathbf{A}$ and $[\mathbf{B}]$. For our tagged linear function family, we define the following sets of scalars, tags, and the domain and range, respectively:

$$\mathcal{S} := \mathbb{Z}_p, \quad \mathcal{T} := \mathbb{G}^{2\times 2}, \quad \mathcal{D} := \mathbb{Z}_p^2, \quad \mathcal{R} := \mathbb{G}^2.$$

Clearly, $\mathcal{D}$ and $\mathcal{R}$ are vector spaces over $\mathcal{S}$. For a tag $[\mathbf{G}] \in \mathbb{G}^{2\times 2}$ and an input $\mathbf{x} \in \mathbb{Z}_p^2$, the tagged linear function $\mathsf{T}_{\mathsf{DDH}}$ is defined as

$$\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{x}) := [\mathbf{Gx}] \in \mathbb{G}^2.$$

We emphasize that the tag $[\mathbf{G}]$ is given in the group, and the domain element $\mathbf{x}$ is given over the field. It is clear that $\mathsf{T}_{\mathsf{DDH}}$ can be computed efficiently and that it is a homomorphism. What remains is to show regularity, translatability and algebraic translation resistance.

Lemma 5.5. *The tagged linear function family* $\mathsf{TLF}_{\mathsf{DDH}}$ *is* ε_{r}*-regular, where* $\varepsilon_{\mathsf{r}} \leq (p+1)/p^2$.

Proof. We define the set Reg from the regularity definition as the set of group parameters and matrices $[\mathbf{T}] \in \mathbb{G}^{2\times 2}$ such that $\mathbf{T} \in \mathbb{Z}_p^{2\times 2}$ is invertible. Then, the probability that a random tag is not in the set is at most $1/p + 1/p^2 = (p+1)/p^2$. This is because for a uniform 2×2 matrix over $\mathbb{Z}_p$, the probability of not being invertible can easily be upper bounded by $1/p^2$, accounting for the chance that the first row is zero, plus $1/p$, accounting for the chance that the second row is a multiple of the first row. Given that the tag is invertible, the distribution of the image of a uniform inputs $\mathbf{x} \xleftarrow{\$} \mathbb{Z}_p^2$ is clearly uniform. □

Lemma 5.6. *The tagged linear function family* $\mathsf{TLF}_{\mathsf{DDH}}$ *is* ε_{t}*-translatable, where* $\varepsilon_{\mathsf{t}} \leq (3+3p)/p^2$.

Proof. To prove the claim, we first have to describe a PPT algorithm Shift and a deterministic polynomial time algorithm Translate. Algorithm Shift takes as input parameters par specifying $\mathbb{G}, p, g$ and a tag $[\mathbf{G}] \in \mathbb{G}^{2\times 2}$. With this input, it is defined as follows:

1. Let $\mathsf{GL}_2(\mathbb{Z}_p)$ be the set of invertible 2×2 matrices over $\mathbb{Z}_p$. Sample a matrix $\mathbf{R} \xleftarrow{\$} \mathsf{GL}_2(\mathbb{Z}_p)$ uniformly at random from that set.
2. Return the tag $[\mathbf{H}] := [\mathbf{RG}] \in \mathbb{G}^{2\times 2}$ and the trapdoor $\mathsf{td} := \mathbf{R}$.

Algorithm $\mathsf{Translate}$ gets as input the trapdoor $\mathsf{td} = \mathbf{R}$ and an element $[\mathbf{y}] \in \mathbb{G}^2$ in the range. It simply outputs $[\mathbf{Ry}] \in \mathbb{G}^2$. Similarly, algorithm $\mathsf{InvTranslate}$ gets as input the trapdoor $\mathsf{td} = \mathbf{R}$ and an element $[\mathbf{y}] \in \mathbb{G}^2$ in the range and outputs $[\mathbf{R}^{-1}\mathbf{y}] \in \mathbb{G}^2$. With this, translation completeness follow easily. For example, we have

$$\begin{aligned}\mathsf{Translate}(\mathbf{R}, \mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{x})) &= \mathsf{Translate}(\mathbf{R}, [\mathbf{Gx}]) \\ &= [\mathbf{RGx}] = [\mathbf{Hx}] = \mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{x}).\end{aligned}$$

It remains to show the well distributed tags property. For that, it is sufficient to bound the statistical distance between

$$\mathcal{T}_0 := \left\{(\mathsf{par}, \mathbf{G}, \mathbf{H}) \;\middle|\; \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda),\ \mathbf{G}, \mathbf{H} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}\right\}$$

and

$$\mathcal{T}_1 := \left\{(\mathsf{par}, \mathbf{G}, \mathbf{H}) \;\middle|\; \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda),\ \mathbf{G} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2},\ \mathbf{R} \xleftarrow{\$} \mathsf{GL}_2(\mathbb{Z}_p),\ \mathbf{H} := \mathbf{RG}\right\}.$$

For $b \in \{0, 1\}$, let $\mathcal{T}_b^*$ denote the distribution $\mathcal{T}_b$ except that all matrices ($\mathbf{G}, \mathbf{H}$ in $\mathcal{T}_0$ and $\mathbf{G}, \mathbf{R}$ in $\mathcal{T}_1$) are sampled uniformly at random from the set $\mathsf{GL}_2(\mathbb{Z}_p)$. We make three observations, finishing the proof:

1. The statistical distance between $\mathcal{T}_0$ and $\mathcal{T}_0^*$ is at most $2\left(1/p + 1/p^2\right)$, because the distributions only differ if at least one of the two random matrices $\mathbf{G}, \mathbf{H}$ is not invertible.
2. The statistical distance between $\mathcal{T}_1$ and $\mathcal{T}_1^*$ is at most $1/p + 1/p^2$, because the distributions only differ if $\mathbf{G}$ is not invertible.
3. The distributions $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ are the same. This is because the set of invertible 2×2 matrices over $\mathbb{Z}_p$ forms a group with respect to multiplication.

□

Lemma 5.7. *Let $t \in \mathbb{N}$ be a number polynomial in λ. If the DDH assumption holds relative to GGen, then $\mathsf{TLF}_{\mathsf{DDH}}$ is t-algebraic translation resistant. Concretely, for any PPT algorithm $\mathcal{A}$ there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{B}) \approx \mathbf{T}(\mathcal{A})$ and*

$$\mathsf{Adv}^{t\text{-A-TRAN-RES}}_{\mathcal{A},\mathsf{TLF}_{\mathsf{DDH}}}(\lambda) \leq \frac{2}{p^2} + \frac{4}{p} + \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda).$$

Proof. Assume that there is an efficient algorithm $\mathcal{A}$ breaking algebraic translation resistance. That is, $\mathcal{A}$ gets as input random tags $[\mathbf{G}], [\mathbf{H}] \xleftarrow{\$} \mathbb{G}^{2\times 2}$ and images $[\mathbf{y}_i] := \mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{x}_i)$ with $\mathbf{x}_i \xleftarrow{\$} \mathbb{Z}_p^2$ for $i \in \{0, \ldots, t\}$. It gets access to the oracle $\textsc{Inv}$ and finally outputs $[\mathbf{y}'_i] \in \mathbb{G}^2$ for all $i \in \{0, \ldots, t\}$. We assume without loss of generality, that $\mathcal{A}$ makes exactly t queries to $\textsc{Inv}$. It wins the game if for all $i \in \{0, \ldots, t\}$ there is a $\mathbf{z}_i \in \mathbb{Z}_p^2$ such that $\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{z}_i) = [\mathbf{y}_i]$ and $\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{z}_i) = [\mathbf{y}'_i]$. The intuition for our proof is that the function $\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \cdot)$ for a random tag $[\mathbf{G}]$ is bijective, so $\mathbf{z}_i = \mathbf{x}_i$ and the adversary will output $\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{x}_i)$. At the same time, based on DDH, the function is indistinguishable from being a compressing function. With an argument similar to what is done in [TZ23], we can then show that $\mathbf{z}_i \neq \mathbf{x}_i$, a contradiction. We now make this intuition formal by providing a sequence of games.

Game $\mathbf{G}_0$: Game $\mathbf{G}_0$ is the algebraic translation resistance game for $\mathsf{TLF}_{\mathsf{DDH}}$ as explained above. By definition, we have

$$\mathsf{Adv}^{\mathsf{A-TRAN-RES}}_{\mathcal{A},\mathsf{TLF}_{\mathsf{DDH}}}(\lambda) = \Pr\left[\mathbf{G}_0 \Rightarrow 1\right].$$

Game $\mathbf{G}_1$: This game is exactly as $\mathbf{G}_0$, but we let the game output 0 if the function defined by the tag $[\mathbf{H}] \in \mathbb{G}^{2\times 2}$ is not bijective. Otherwise, the game behaves as the previous game. More precisely, the game no longer samples $[\mathbf{H}] \xleftarrow{\$} \mathbb{G}^{2\times 2}$ but instead samples $\mathbf{H} \xleftarrow{\$} \mathbb{Z}_p^{2\times 2}$ and checks if $\mathbf{H}$ is invertible. If it is not, the game outputs 0. Otherwise, the game continues as in $\mathbf{G}_0$ using tag $[\mathbf{H}]$. Clearly, the distribution of $[\mathbf{H}]$ did not change. The probability of the matrix not being invertible can easily be upper bounded by $1/p^2$, accounting for the chance that the first row is zero, plus $1/p$, accounting for the chance that the second row is a multiple of the first row. We get

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{1}{p^2} + \frac{1}{p}.$$

Game $\mathbf{G}_2$: In this game, we change the winning condition. Recall that until now, the game outputs 1 if for all $i \in \{0,\dots,t\}$ there is a $\mathbf{z}_i \in \mathbb{Z}_p^2$ such that $\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{z}_i) = [\mathbf{y}_i]$ and $\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{z}_i) = [\mathbf{y}'_i]$. Now, we replace this condition as follows: For each $i \in \{0,\dots,t\}$, the game first computes

$$[\mathbf{w}_i] := [\mathbf{H}^{-1}\mathbf{y}'_i] \in \mathbb{G}^2.$$

Observe that $\mathbf{H}^{-1}$ exists and the game holds it due to the previous change, and thus the game can efficiently compute $[\mathbf{w}_i]$ over the group. Further, observe that $\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{w}_i) = [\mathbf{y}'_i]$. Given these $[\mathbf{w}_i]$, the game then accepts if and only if for all $i \in \{0,\dots,t\}$, we have $[\mathbf{w}_i] = [\mathbf{x}_i]$. In other words, the game checks that $\mathbf{w}_i = \mathbf{x}_i$, which it can do efficiently in the exponent. We argue that, except with negligible probability, the winning condition is equivalent to the winning condition in the previous game. Namely, except with probability $1/p^2 + 1/p$, the function $\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \cdot)$ is a bijection. This can be seen with arguments similar to the previous game. Assuming that it is a bijection, we now argue that the two winning conditions are equivalent:

- If the new winning condition in $\mathbf{G}_2$ holds, we can set $\mathbf{z}_i = \mathbf{w}_i = \mathbf{x}_i$ for all $i \in \{0,\dots,t\}$, which shows that the old winning condition in $\mathbf{G}_1$ holds.
- If the old winning condition in $\mathbf{G}_1$ holds, then we know that for all $i \in \{0,\dots,t\}$, we have

  $$\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{z}_i) = [\mathbf{y}'_i] = \mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \mathbf{w}_i).$$

  As $\mathsf{T}_{\mathsf{DDH}}([\mathbf{H}], \cdot)$ is a bijection, this means that $\mathbf{z}_i = \mathbf{w}_i$. Thus, we have

  $$\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{w}_i) = \mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{z}_i) = [\mathbf{y}_i] = \mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{x}_i),$$

  where the last equality follows from the old winning condition. As we assume that $\mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \cdot)$ is a bijection, this implies $\mathbf{w}_i = \mathbf{x}_i$, showing that the new winning condition holds.

With that, we obtain

$$|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_2 \Rightarrow 1]| \leq \frac{1}{p^2} + \frac{1}{p}.$$

Game $\mathbf{G}_3$: This game is exactly as $\mathbf{G}_2$, but we change the way the tag $\mathbf{G} \in \mathbb{G}^{2\times 2}$ is generated. Intuitively, while $\mathbf{G}$ induced a bijection (except with negligible probability) before, we now make sure that it induces a compressing function. Namely, instead of sampling $[\mathbf{G}]$ uniformly at random, we generate it as

$$[\mathbf{G}] := \begin{pmatrix} g^{\beta} & h \\ g^{\alpha\beta} & h^{\alpha} \end{pmatrix} \text{ for } h \xleftarrow{\$} \mathbb{G},\ \alpha, \beta \xleftarrow{\$} \mathbb{Z}_p.$$

We can bound the distinguishing advantage between games $\mathbf{G}_2$ and $\mathbf{G}_3$ using a straight-forward reduction $\mathcal{B}$ against the DDH assumption.

1. The reduction $\mathcal{B}$ gets as input parameters defining the group $\mathbb{G}$ with a generator g, a random element $h \in \mathbb{G}$ and group elements $u, v \in \mathbb{G}$, which are either both random or of the form $u = g^{\alpha}, v = h^{\alpha}$.

2. The reduction $\mathcal{B}$ sets up the tag $[\mathbf{H}]$ and the vectors $\mathbf{x}_i$ as in $\mathbf{G}_2$. Then, it samples $\beta \xleftarrow{\$} \mathbb{Z}_p$ and defines

$$[\mathbf{G}] := \begin{pmatrix} g^\beta & h \\ u^\beta & v \end{pmatrix}.$$

   With this, it computes $[\mathbf{y}_i] := \mathsf{T}_{\mathsf{DDH}}([\mathbf{G}], \mathbf{x}_i)$ as in $\mathbf{G}_2$ and runs $\mathcal{A}$, simulating the oracle Inv for $\mathcal{A}$ as in $\mathbf{G}_2$.

3. When $\mathcal{A}$ terminates and outputs $[\mathbf{y}'_i] \in \mathbb{G}^2$ for all $i \in \{0, \ldots, t\}$, the reduction checks the winning condition as in $\mathbf{G}_2$. Recall that this can be done efficiently, due to the change in $\mathbf{G}_2$. It outputs whatever $\mathbf{G}_2$ would output.

Clearly, the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. Further, if $\mathcal{B}$'s input is random, $\mathcal{B}$ perfectly simulates $\mathbf{G}_2$ for $\mathcal{A}$ unless $\beta = 0$, which happens with probability $1/p$. On the other hand, if the input is of the form $u = g^\alpha, v = h^\alpha$, then $\mathcal{B}$ perfectly simulates $\mathbf{G}_3$ for $\mathcal{A}$. We get

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{B},\mathsf{GGen}}(\lambda) + \frac{1}{p}.$$

What remains is to show that the probability that $\mathbf{G}_3$ outputs 1 is negligible. The intuition is that now $\mathbf{G}$ has low rank, and $\mathcal{A}$ does not obtain enough information about $\mathbf{x}_0, \ldots, \mathbf{x}_t$. To turn this intuition into a formal argument, we first introduce more notation.

Notation. We set $\mathbf{X} \in \mathbb{Z}_p^{2\times(t+1)}$ to be the matrix with columns $\mathbf{x}_i$, $\mathbf{Y} := \mathbf{G}\mathbf{X}$ to be the matrix with columns $\mathbf{y}_i$, and $\mathbf{Y}' \in \mathbb{Z}_p^{2\times(t+1)}$ be the matrix with columns $\mathbf{y}'_i$. The winning condition introduced in $\mathbf{G}_2$ can now be written as $\mathbf{H}^{-1}\mathbf{Y}' = \mathbf{X}$.

Claim. We claim that for all fixed parameters par, each fixed $\mathbf{X}$, and for fixed random coins ρ for $\mathcal{A}$, there is a matrix $\mathbf{K} \in \mathbb{Z}_p^{2\times(t+1)} \setminus \{\mathbf{0}\}$ such that the view of $\mathcal{A}$ in $\mathbf{G}_3$ is independent of $\vartheta \in \mathbb{Z}_p$ if we replace $\mathbf{X}$ with $\mathbf{X} + \vartheta\mathbf{K}$. Assuming this claim holds, it is clear that

$$\Pr[\mathbf{G}_3 \Rightarrow 1] \leq \frac{1}{p}.$$

Proof of Claim. It remains to show the claim. Fix parameters par, a matrix $\mathbf{X}$, and random coins ρ. Observe that with that, all t queries of $\mathcal{A}$ to Inv are defined. Let the jth query to this oracle be $\textsc{Inv}(\alpha_0^{(j)}, \ldots, \alpha_t^{(j)})$. The queries define a matrix $\mathbf{A} \in \mathbb{Z}_p^{(t+1)\times t}$ where the jth column of $\mathbf{A}$ is $(\alpha_0^{(j)}, \ldots, \alpha_t^{(j)})$. With this in mind, it is clear that the view of $\mathcal{A}$ in game $\mathbf{G}_3$ is given by

$$\mathbf{H},\ \mathbf{G},\ \mathbf{G}\mathbf{X},\ \mathbf{X}\mathbf{A}.$$

Therefore, it is sufficient to show that there is a matrix $\mathbf{K} \in \mathbb{Z}_p^{2\times(t+1)} \setminus \{\mathbf{0}\}$ such that

$$\mathbf{G}\mathbf{K} = \mathbf{0} \text{ and } \mathbf{K}\mathbf{A} = \mathbf{0}.$$

By the way we sample $\mathbf{G}$ in game $\mathbf{G}_3$, there exists a vector $\mathbf{g}_\perp \in \mathbb{Z}_p^2 \setminus \{\mathbf{0}\}$ such that $\mathbf{G}\mathbf{g}_\perp = \mathbf{0}$. Also, there is a vector $\mathbf{a}_\perp \in \mathbb{Z}_p^{t+1} \setminus \{\mathbf{0}\}$ such that $\mathbf{a}_\perp^t \mathbf{A} = \mathbf{0}$, which follows from the dimensions of $\mathbf{A}$. Now, we can set

$$\mathbf{K} := \mathbf{g}_\perp \mathbf{a}_\perp^t \in \mathbb{Z}_p^{2\times(t+1)} \setminus \{\mathbf{0}\}.$$

□

5.6 Concrete Parameters and Efficiency

In the final section of this chapter, we briefly discuss the concrete efficiency of our threshold signature schemes. We compare our constructions to Frost [KG20, BTZ22, BCK+22], TZ [TZ23], and Sparkle [CKM23a]. For our comparison, we assume that all constructions are instantiated with the secp256k1 curve and SHA-256 as a hash function. We calculate key sizes, communication complexity per signer, and signature sizes. Also, for all schemes, we assume challenges c_i are sampled uniformly from $\mathbb{Z}_p$, i.e., have size 256 bit, whereas some implementations may use challenges of size 128 bit. Our results have been computed using a simple Python script[7] and are summarized in Table 5.2. We see that

Scheme	Public Key	Communication	Signature
Frost	33	98	64
TZ	33	130	97
Sparkle	33	97	64
Twinkle (AOMCDH)	33	163	97
Twinkle (DDH)	66	294	162

Table 5.2: Concrete efficiency of threshold signature schemes in the discrete logarithm setting without pairings. We compare the size of public keys, the communication complexity per signer, and the signature size. Sizes are given in bytes.

the sizes of our constructions are practical, but slightly less efficient compared to previous schemes. In terms of computation, consider a run of the signing protocol in which K signers participate. Here, a signer in both of our schemes has to evaluate $K + 2$ hashes, whereas it would have to evaluate $K + 1$ hashes in Sparkle, which is a minimal difference as K grows. A signer in Sparkle would have to compute 2 group operations (counting multi-scalar multiplications as one operation), whereas a signer in our AOMCDH-based (resp. DDH-based) scheme would have to do 6 (resp. 12) such operations. For verification, the number of operations compared to Sparkle increases by a factor of 2 for the hashes, and a factor of 2 (resp. 4) for the multi-scalar multiplications for our AOMCDH-based (resp. DDH-based) scheme.

To sum up, our schemes are slightly less efficient than previous schemes, but they are still in a highly practical regime. Given the strong properties that our schemes achieve from conservative assumptions without the algebraic group model, it is natural to pay such a small price in terms of efficiency.

[7] The Python script can be found in https://github.com/b-wagn/dissertation-efficiency-scripts.

6
Final Remarks

6.1 Open Problems for Future Work

At the end of this dissertation, we motivate and state some open problems pertaining our results. This may serve as a point of reference for future research endeavors.

6.1.1 Open Problems Related to Blind Signatures – Chapter 3

Reflecting on our constructions of blind signatures in Chapter 3, one way to interpret the cut-and-choose technique is the following: we let the User and the Signer run the underlying base protocol, e.g., blind BLS [Bol03], and make the User prove to the Signer that it behaved honestly. In other words, the cut-and-choose informally served as a zero-knowledge proof. Crucially, utilizing generic zero-knowledge proofs is impractical in this scenario. The relation to be proven is defined by a random oracle, and employing generic proofs would necessitate treating the random oracle as a circuit, a non-standard approach with unclear security implications. Motivated by this observation, one may try to apply the cut-and-choose technique in other contexts where proofs about such random oracle relations are needed. On a broader scale, we would hope to define an abstract framework for proofs concerning such relations and then construct a general-purpose proof using cut-and-choose. This yields the following open problem.

Open Problem 1. *Use the (parallel instance) cut-and-choose technique developed in Chapter 3 to construct generic proofs about relations defined with respect to a random oracle.*

Our most efficient construction of blind signatures presented in Chapter 3 is heavily reliant on pairings. However, as discussed in Chapter 5, there is a compelling motivation to explore constructions in pairing-free cyclic groups: operations in pairing-free groups are more efficient, assumptions in this setting are weaker, and there is broader library support available. As explained in Chapter 3, instantiating single instance cut-and-choose with the pairing-free base schemes such as Schnorr blind signatures [Sch91, CP93, Bra94] or Okamoto-Schnorr blind signatures [Oka93, PS00, HKL19] would require impractical parameters. On the other hand, parallel instance cut-and-choose fails to be efficient due to suboptimal aggregation features of these base schemes. In our work, the construction that comes closest to the goal of efficient pairing-free blind signatures is our RSA-based construction. This construction is pairing-free but operates outside the realm of cyclic groups and has a stateful signer. Additionally, a recent construction by Chairattana-Apirom, Tessaro, and Zhu [CATZ23] builds on our ideas from Chapter 3 to construct efficient pairing-free blind signatures from CDH. However, it only achieves a weaker variant of one-more unforgeability.

Open Problem 2. *Construct an efficient and fully secure blind signature scheme from well-studied non-interactive assumptions in the pairing-free discrete logarithm setting.*

In certain scenarios, the signature verification algorithm is predetermined and we need to design a blind signing protocol that generates these signatures. Consider, for instance, the issuance of blind signatures on Bitcoin transactions. Here, the signatures should be Schnorr signatures [Sch91], as the consensus rules on the blockchain do not allow for a custom signature verification algorithm. Such Schnorr-compatible signing protocols have been studied extensively for other signature variants, e.g., [KG20, BDN18, MPSW19, NRSW20, NRS21, BCK+22, RRJ+22], but no such protocol with concurrent security for blind signatures is known[1]. As explained earlier, instantiating the (parallel instance) cut-and-choose technique from Chapter 3 with Schnorr blind signatures leads to an inefficient scheme. In addition, the resulting signatures are not Schnorr-compatible: for example, they contain the commitment randomness of the unopened sessions.

[1]The only exception seems to be the work by Fuchsbauer and Wolf [FW22], which uses generic zero-knowledge proofs and assumes that Schnorr signatures are secure for a fixed non-random oracle hash function.

Open Problem 3. *Construct an efficient and fully secure blind signature scheme from well-studied non-interactive assumptions, such that verification is the same as in widely-deployed signature schemes, e.g., BLS [BLS01] or Schnorr [Sch91].*

6.1.2 Open Problems Related to Multi-Signatures – Chapter 4

In Chapter 4, we developed techniques to construct pairing-free two-round multi-signatures that do not rely on rewinding. This led to two flavors of constructions: one with tight security and one with key aggregation, but with a linear security loss in the number of signing queries. Naturally, the question whether we can have the best of the two worlds arises: we wish for a scheme with both key aggregation and a tight security proof. To understand the intricacy of this challenge, assume that we want to aggregate keys for the tightly secure construction in Toothpicks. In this setting, recall that each signer is endowed with two keys and selects the key to use for each message pseudorandomly. Consequently, for N signers, there are 2^N possible combinations of keys, and one of them is chosen pseudorandomly during the signing process. Thus, we would have to include all 2^N keys in the aggregate key, which is much larger than just storing the N individual keys.

Open Problem 4. *Construct a tightly-secure two-round multi-signature scheme in the pairing-free discrete logarithm setting that supports key aggregation.*

Our constructions in Chapter 4 rely on a variant of the lossy identification technique [KW03, AFLT12, KMP16] and thus on a decisional assumption, namely, DDH. A natural avenue for exploration is whether we can design a scheme with comparable guarantees from a weaker search assumption, e.g., CDH. Even though the techniques in Chapters 4 and 5 both rely on the DDH assumption, we envision that combining them is a good start. Namely, recall that the established security model for multi-signatures also used in Chapter 4 assumes that there is one fixed honest party that is controlled by the game. The adversary controls all other parties and has to forge a signature with respect to a set of parties including the honest party. For multi-signatures and standard signatures, this model of static corruptions generically implies security under adaptive corruptions via a simple guessing argument, which is in contrast to threshold signatures. Our security analysis for threshold signatures in Chapter 5 initially employed a one-more search assumption, calling an oracle for each corruption query, and subsequently utilized DDH to simulate this assumption. However, for multi-signatures, where we do not need an adaptive corruption oracle, the use of a one-more assumption and, consequently, DDH may be avoidable. This would ideally result in a three-move scheme based on CDH. We can then try to use techniques from Chapter 4 to achieve tightness and reduce the number of rounds.

Open Problem 5. *Construct a tightly-secure two-round multi-signature scheme in the pairing-free discrete logarithm setting based on a non-interactive search assumption, e.g.,* CDH*, potentially combining some of the techniques from Chapters 4 and 5.*

As already explained above, adaptive security for multi-signatures and standard signatures is generically implied by static security. Namely, a reduction can guess which of the initially honest parties remains uncorrupted for the forgery. From the concrete security perspective, however, achieving adaptive security directly would be more desirable, as the guessing argument used to prove the implication incurs a security loss. The loss is linear in the number of initially honest parties which can be corrupted adaptively. Consequently, we want an efficient scheme that is tightly secure in the adaptive setting. Analogous questions have been explored for standard digital signatures [BHJ+15, GJ18, HJK+21, DGJL21, PW22] but not for multi-signatures. Combining our techniques from Chapters 4 and 5 seems again promising in this regard: we supported adaptive corruptions in Chapter 5 and we achieved tightness in Chapter 4. Also, the avoidance of rewinding in both constructions aligns with the objective of tight security.

Open Problem 6. *Construct a multi-signature scheme that is tightly secure in presence of adaptive corruptions, potentially combining some of the techniques from Chapters 4 and 5.*

6.1.3 Open Problems Related to Threshold Signatures – Chapter 5

In the preceding section, we presented potential applications of merging techniques introduced in Chapters 4 and 5 in the realm of multi-signatures. Indeed, these techniques seem to exhibit sufficient structural compatibility for that. Turning our attention to the threshold setting, a natural next step is reducing the number of rounds from three to two. In our threshold signature scheme in Chapter 5, an initial round of random oracle commitments to the nonces precedes the actual protocol. Interestingly, in Chapter 4 we successfully used homomorphic trapdoor commitments to eliminate the need for such an extra round. Thus, we may try to adopt our commitment construction from Chapter 4 to align with our threshold signature scheme in Chapter 5. Similarly, avoiding the guessing argument in the proof in Chapter 5 to get a tight proof is desirable. For that, the pseudorandom path technique from Chapter 4 can serve as a starting point.

Open Problem 7. *Construct a two-round pairing-free threshold signature scheme for up to t adaptive corruptions, potentially combining some of the techniques from Chapters 4 and 5.*

Open Problem 8. *Construct a threshold signature scheme that is tightly secure in presence of up to t adaptive corruptions, potentially combining some of the techniques from Chapters 4 and 5.*

As for multi-signatures, an interesting direction to explore is whether we can further weaken the hardness assumptions used in our threshold signature scheme. Again, a construction based on a non-interactive search assumption such as CDH would be desirable. Note, however, that this problem seems much more intricate than the respective problem for multi-signatures because we have to simulate adaptive corruptions in the threshold setting. Achieving this without leveraging interactive or decisional assumptions requires fundamentally new insights.

Open Problem 9. *Construct a pairing-free threshold signature scheme for up to t adaptive corruptions from a non-interactive search assumption, e.g.,* CDH.

We have made a significant contribution by constructing the first threshold signature scheme in the pairing-free setting tolerating up to t adaptive corruptions without the algebraic group model. Reflecting on our security model, we have disallowed the adversary from initiating any signing session concerning the forgery message. As outlined in Chapter 5, this restriction is common for interactive threshold signatures. However, following the overarching theme of this dissertation, we may try to further strengthen the model by allowing the adversary to start such signing sessions to some extent. For example, for non-interactive threshold signatures, e.g., [Bol03, BL22], the adversary is allowed to see up to t signature shares for the forgery message. Defining a signature share in the realm of interactive threshold signatures poses challenges. We may, for example, allow the adversary to communicate with all signers in a signing session for the forgery message, as long as the final message of at most t signers is given to the adversary. An even more robust notion would permit the adversary to do this in multiple different signing sessions. For our construction in Chapter 5, we are confident that we can allow the adversary to learn up to t messages during the second signing round. We decided not to define and prove such an intermediate notion and instead opted for a clean model. Designing a scheme with analogous properties, e.g., pairing-free, adaptive corruptions, that supports even more communication for the forgery message is an interesting direction. Such a scheme would inherently be more resilient in practical applications.

Open Problem 10. *Construct a pairing-free threshold signature scheme for up to t adaptive corruptions and for which the adversary is allowed to interact with honest signers for the forgery message.*

6.2 Conclusion

In this dissertation, we have directed our attention towards modern variants of digital signatures, specifically blind signatures, multi-signatures, and threshold signatures. We have developed new techniques to obtain constructions that are both efficient and secure.

Our achievements for blind signatures build on a variant of the cut-and-choose technique. This technique allows us to turn a base scheme with a weak security property into a fully secure scheme. Intuitively, it forces a malicious user to follow the protocol, up to a small number of successful cheats. By incorporating several enhancements, we have made an exponential improvement to the communication complexity of this transformation, and have been able to start from an even weaker security property. Exploiting the structure of the base scheme, coupled with several other optimizations, we have ensured that our scheme is not only asymptotically, but also concretely efficient. In a second result, we have then made the Signer stateless and the scheme both round-optimal and computationally efficient.

With regards to two-round multi-signatures, we have circumvented the need for rewinding in the security proof, resulting in a substantial improvement of the concrete security bound. At the same time, we have managed to prove our schemes from a non-interactive assumption. Our main technique was a novel homomorphic commitment scheme with a relaxed equivocation property in combination with a lossy identification scheme. Building on this first result for multi-signatures, we have then mitigated the efficiency overhead of our constructions, for example by also relaxing the binding property of the commitment scheme.

In the realm of threshold signatures with adaptive security, we have replicated our success in avoiding rewinding in the security proof and eliminating the reliance on interactive assumptions. Notably, while these technical goals parallel those of multi-signatures, the motivations and techniques employed differ significantly. Namely, the main motivation for avoiding rewinding when constructing threshold signatures was to overcome an unnatural constraint on the number of adaptive corruptions as present in previous works. In terms of techniques, we have found that the lossy identification technique is not applicable when handling adaptive corruptions. Instead, our threshold signature scheme builds on five-move identification. Also, while we have directly avoided interactive assumptions using our special commitment in the case of multi-signatures, we have followed a more indirect yet structured approach for threshold signatures: after designing our scheme abstractly based on an interactive assumption, we have developed an instantiation for which this interactive assumption is implied by a well-studied non-interactive assumption.

The techniques developed in this dissertation and summarized above have resulted in notable advancements surpassing the current state-of-the-art in blind signatures, multi-signatures, and threshold signatures. In addition, they may lay a foundation for future research, for example in addressing the challenges listed in the open problems section. I am also confident that the techniques possess a versatility that extends beyond the specific domains of blind signatures, multi-signatures, and threshold signatures. They have the potential to be adapted and applied for different signature variants or even broader cryptographic contexts. Concluded.

Bibliography

[AB21] Handan Kilinç Alper and Jeffrey Burdges. Two-round trip schnorr multi-signatures via delinearized witnesses. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 157–188, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84242-0_7.
Cited on Page 83, 135.

[Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44987-6_9.
Cited on Page 21.

[ABP13] Michel Abdalla, Fabrice Ben Hamouda, and David Pointcheval. Tighter reductions for forward-secure signature schemes. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 292–311, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-36362-7_19.
Cited on Page 4.

[ADN06] Jesús F. Almansa, Ivan Damgård, and Jesper Buus Nielsen. Simplified threshold RSA with adaptive and proactive security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 593–611, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_35.
Cited on Page 135.

[AEE+21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 635–664, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92075-3_22.
Cited on Page 4.

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 244–251, Kyongju, Korea, November 3–7, 1996. Springer, Heidelberg, Germany. doi:10.1007/BFb0034851.
Cited on Page 34.

[AFLT12] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 572–590, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_34.
Cited on Page 6, 8, 84, 85, 88, 91, 97, 115, 136, 142, 164.

[AGH10] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. Synchronized aggregate signatures: new definitions, constructions and applications. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010: 17th Conference on Computer and Communications Security*, pages 473–484, Chicago, Illinois, USA, October 4–8, 2010. ACM Press. doi:10.1145/1866307.1866360.
Cited on Page 87.

[AHS20] Jean-Philippe Aumasson, Adrian Hamelink, and Omer Shlomovits. A survey of ECDSA threshold signing. Cryptology ePrint Archive, Report 2020/1390, 2020. https://eprint.iacr.org/2020/1390.
Cited on Page 135.

[AKSY21] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Can round-optimal lattice-based blind signatures be practical? Cryptology ePrint Archive, Report 2021/1565, 2021. https://eprint.iacr.org/2021/1565.
Cited on Page 24.

[AO00] Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 271–286, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-44598-6_17.
Cited on Page 21, 34.

[ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *ICALP 2022: 49th International Colloquium on Automata, Languages and Programming*, volume 229 of *LIPIcs*, pages 8:1–8:20, Paris, France, July 4–8, 2022. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ICALP.2022.8.
Cited on Page 135.

[BBDP22] Zvika Brakerski, Pedro Branco, Nico Döttling, and Sihang Pu. Batch-OT with optimal rate. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 157–186, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-07085-3_6.
Cited on Page 24, 34.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_18.
Cited on Page 6.

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_11.
Cited on Page 6.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-28628-8_3.
Cited on Page 4.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 449–458, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press. doi:10.1145/1455770.1455827.
Cited on Page 84.

[BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 517–550, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15985-5_18.
Cited on Page 134, 140, 159, 163.

[BD21] Mihir Bellare and Wei Dai. Chain reductions for multi-signatures and the HBMS scheme. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 650–678, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92068-5_22.
Cited on Page 83, 85, 86, 87, 135.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 435–464, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-03329-3_15.
Cited on Page 4, 83, 85, 86, 88, 135, 163.

[BFM15] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, pages 428–455, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46497-7_17.
Cited on Page 6.

[BFP21] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. In Mehdi Tibouchi and Huaxiong

Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 587–617, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92068-5_20.
Cited on Page 16.

[BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 565–596, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96884-1_19.
Cited on Page 135.

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 1–16, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_1.
Cited on Page 79.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-39200-9_26.
Cited on Page 4, 85, 87.

[BHJ+15] Christoph Bader, Dennis Hofheinz, Tibor Jager, Eike Kiltz, and Yong Li. Tightly-secure authenticated key exchange. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 629–658, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46494-6_26.
Cited on Page 6, 164.

[BHK+23] Fabrice Benhamouda, Shai Halevi, Hugo Krawczyk, Yiping Ma, and Tal Rabin. Sprint: High-throughput robust distributed schnorr signatures. Cryptology ePrint Archive, Report 2023/427, 2023. https://eprint.iacr.org/2023/427.
Cited on Page 135.

[BJLS16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. On the impossibility of tight cryptographic reductions. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 273–304, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49896-5_10.
Cited on Page 6.

[BKKP15] Olivier Blazy, Saqib A. Kakvi, Eike Kiltz, and Jiaxin Pan. Tightly-secure signatures from chameleon hash functions. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 256–279, Gaithersburg, MD, USA, March 30 – April 1,

2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46447-2__12.
Cited on Page 6.

[BKM06] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 60–79, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. doi:10.1007/11681878__4.
Cited on Page 4.

[BKP13] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 218–236, Banff, AB, Canada, June 25–28, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-38980-1__14.
Cited on Page 135.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 408–425, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-44371-2__23.
Cited on Page 6.

[BKP20] Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 464–492, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64834-3__16.
Cited on Page 4.

[BL13a] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013: 20th Conference on Computer and Communications Security*, pages 1087–1098, Berlin, Germany, November 4–8, 2013. ACM Press. doi:10.1145/2508859.2516687.
Cited on Page 5.

[BL13b] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 82–99, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-42045-0__5.
Cited on Page 24.

[BL16] Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 404–434, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53890-6__14.
Cited on Page 6.

[BL22] Renas Bacho and Julian Loss. On the adaptive security of the threshold BLS signature scheme. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 193–207, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560656.
Cited on Page 135, 165.

[BLL+21] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 33–53, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-77870-5_2.
Cited on Page 21.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-45682-1_30.
Cited on Page 3, 7, 24, 27, 46, 47, 69, 79, 135, 164.

[BLT+23] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from ddh with full adaptive security. Cryptology ePrint Archive, Paper 2023/1482, 2023. https://eprint.iacr.org/2023/1482. URL: https://eprint.iacr.org/2023/1482.
Cited on Page 16, 132.

[BLT+24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from ddh with full adaptive security. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 429–459, Zurich, Switherland, May 26–30, 2024. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-58716-0_15.
Cited on Page vii, 6, 9, 16, 132.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48405-1_28.
Cited on Page 4.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-39200-9_38.
Cited on Page 4.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 390–399, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. doi:10.1145/1180405.1180453.
Cited on Page 4, 83, 85, 86, 88, 94, 95, 135, 137.

[BNN09] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, January 2009. doi:10.1007/s00145-008-9028-8.
Cited on Page 4.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003. doi:10.1007/s00145-002-0120-1.
Cited on Page 21, 24.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-36288-6_3.
Cited on Page 7, 21, 22, 23, 24, 27, 28, 42, 83, 85, 135, 163, 165.

[BP22] Luís T. A. N. Brandão and Rene Peralta. NIST IR 8214C: First call for multi-party threshold schemes. https://csrc.nist.gov/pubs/ir/8214/c/ipd, 2022. Accessed: 2023-09-12.
Cited on Page 133.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. doi:10.1145/168588.168596.
Cited on Page 6, 14, 83.

[Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 302–318, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Heidelberg, Germany. doi:10.1007/3-540-48329-2_26.
Cited on Page 163.

[BTT22] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. MuSig-L: Lattice-based multi-signature with single-round online phase. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 276–305, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15979-4_10.
Cited on Page 83, 85, 135.

[BTZ22] Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: BLS and FROST. Cryptology ePrint Archive, Report 2022/833, 2022. https://eprint.iacr.org/2022/833.
Cited on Page 134, 140, 159.

[CAHL+22a] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 3–31, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

doi:10.1007/978-3-031-15982-4_1.
Cited on Page vii, 6, 9, 20, 23, 25, 26, 27, 28, 29, 97, 134.

[CAHL+22b] Rutchathon Chairattana-Apirom, Lucjan Hanzlik, Julian Loss, Anna Lysyanskaya, and Benedikt Wagner. PI-cut-choo and friends: Compact blind signatures via parallel instance cut-and-choose and more. Cryptology ePrint Archive, Report 2022/007, 2022. https://eprint.iacr.org/2022/007.
Cited on Page 20, 53.

[CAL22] Rutchathon Chairattana-Apirom and Anna Lysyanskaya. Compact cut-and-choose: Boosting the security of blind signature schemes, compactly. Cryptology ePrint Archive, Report 2022/003, 2022. https://eprint.iacr.org/2022/003.
Cited on Page 20, 27.

[Cam97] Jan Camenisch. Efficient and generalized group signatures. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 465–479, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. doi:10.1007/3-540-69053-0_32.
Cited on Page 4.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000. doi:10.1007/s001459910006.
Cited on Page 24.

[CATZ23] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. Pairing-free blind signatures from cdh assumptions. Cryptology ePrint Archive, Report 2023/1780, 2023. https://eprint.iacr.org/2023/1780.
Cited on Page 25, 163.

[CCL+20] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold EC-DSA. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 266–296, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45388-6_10.
Cited on Page 135.

[CG08] Jan Camenisch and Thomas Groß. Efficient attributes for anonymous credentials. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008: 15th Conference on Computer and Communications Security*, pages 345–356, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press. doi:10.1145/1455770.1455814.
Cited on Page 3, 21.

[CGG+20] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1769–1787, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3423367.
Cited on Page 135.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, TX, USA, May 23–26, 1998. ACM Press. doi:10.1145/276698.276741.
Cited on Page 6.

[CGJ+99] Ran Canetti, Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 98–115, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48405-1_7.
Cited on Page 134, 135.

[CGRS23] Hien Chu, Paul Gerhart, Tim Ruffing, and Dominique Schröder. Practical schnorr threshold signatures without the algebraic group model. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 743–773, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38557-5_24.
Cited on Page 134, 140.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*, volume 4596 of *Lecture Notes in Computer Science*, pages 423–434, Wroclaw, Poland, July 9–13, 2007. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-73420-8_38.
Cited on Page 4.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.
Cited on Page 3, 21.

[Che05] Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 511–526, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. doi:10.1007/11535218_31.
Cited on Page 134, 135, 136.

[Che23] Yanbo Chen. DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 716–747, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38554-4_23.
Cited on Page 85.

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two pairing-based aggregate signature schemes. *Des. Codes Cryptogr.*, 55(2-3):141–167, 2010. URL: https://doi.org/10.1007/s10623-009-9334-7, doi:10.1007/S10623-009-9334-7.
Cited on Page 18.

[CKM21] Elizabeth Crites, Chelsea Komlo, and Mary Maller. How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. Cryptology ePrint Archive, Report 2021/1375, 2021. https://eprint.iacr.org/2021/1375.
Cited on Page 83, 94, 134.

[CKM23a] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 678–709, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38557-5_22.
Cited on Page 8, 133, 134, 135, 136, 140, 159.

[CKM+23b] Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Snowblind: A threshold blind signature in pairing-free groups. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part I*, volume 14081 of *Lecture Notes in Computer Science*, pages 710–742, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38557-5_23.
Cited on Page 4, 133.

[CKP+23] Elizabeth Crites, Markulf Kohlweiss, Bart Preneel, Mahdi Sedaghat, and Daniel Slamanig. Threshold structure-preserving signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part II*, volume 14439 of *Lecture Notes in Computer Science*, pages 348–382, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-981-99-8724-5_11.
Cited on Page 135.

[CKS05] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in Constantinople: Practical asynchronous byzantine agreement using cryptography. *Journal of Cryptology*, 18(3):219–246, July 2005. doi:10.1007/s00145-005-0318-0.
Cited on Page 4.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44987-6_7.
Cited on Page 3, 21.

[CLW24] Cas Cremers, Julian Loss, and Benedikt Wagner. A holistic security analysis of monero transactions. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part III*, volume 14653 of *Lecture Notes in Computer Science*, pages 129–159, Zurich, Switherland, May 26–30, 2024. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-58734-4_5.
Cited on Page vii.

[CM99] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 107–122, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. doi:10.1007/3-540-48910-X_8.
Cited on Page 22, 63.

[CM09] Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings – the role of Ψ revisited. Cryptology ePrint Archive, Report 2009/480, 2009. https://eprint.iacr.org/2009/480.
Cited on Page 79.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 89–105, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. doi:10.1007/3-540-48071-4_7.
Cited on Page 127, 163.

[CP06] Richard Crandall and Carl B Pomerance. *Prime numbers: a computational perspective*, volume 182. Springer Science & Business Media, 2006.
Cited on Page 60.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany. doi:10.1007/3-540-46416-6_22.
Cited on Page 4.

[CW13] Jie Chen and Hoeteck Wee. Fully, (almost) tightly secure IBE and dual system groups. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 435–460, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_25.
Cited on Page 6.

[Dam00] Ivan Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_30.
Cited on Page 88.

[DCX+23] Sourav Das, Philippe Camacho, Zhuolun Xiang, Javier Nieto, Benedikt Bunz, and Ling Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. Cryptology ePrint Archive, Report 2023/598, 2023. https://eprint.iacr.org/2023/598.
Cited on Page 135.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th Annual ACM Symposium on Theory of Computing*, pages 522–533, Montréal, Québec, Canada, May 23–25, 1994. ACM Press. doi:10.1145/195058.195405.
Cited on Page 135.

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bryan Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igors Stepanovs. On the security of two-round multi-signatures. In *2019 IEEE Symposium on Security and Privacy*, pages 1084–1101, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press. doi:10.1109/SP.2019.00050.
Cited on Page 83, 84, 89.

[Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 120–127, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany. doi:10.1007/3-540-48184-2_8.
Cited on Page 4, 87, 133.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34805-0_28.
Cited on Page 4, 87, 133.

[DG21] Hannah Davis and Felix Günther. Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In Kazue Sako and Nils Ole Tippenhauer, editors, *ACNS 21: 19th International Conference on Applied Cryptography and Network Security, Part II*, volume 12727 of *Lecture Notes in Computer Science*, pages 448–479, Kamakura, Japan, June 21–24, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-78375-4_18.
Cited on Page 6.

[DGJL21] Denis Diemert, Kai Gellert, Tibor Jager, and Lin Lyu. More efficient digital signatures with tight multi-user security. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12711 of *Lecture Notes in Computer Science*, pages 1–31, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75248-4_1.
Cited on Page 6, 164.

[DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In Srdjan Capkun and Franziska Roesner, editors, *USENIX Security 2020: 29th USENIX Security Symposium*, pages 2093–2110. USENIX Association, August 12–14, 2020.
Cited on Page 85.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. doi:10.1109/TIT.1976.1055638.
Cited on Page 3, 6, 15, 16, 17.

[DHP23] Khue Do, Lucjan Hanzlik, and Eugenio Paracucchi. M&m's: Mix and match attacks on schnorr-type blind signatures with repetition. Cryptology ePrint Archive, Report 2023/1588, 2023. https://eprint.iacr.org/2023/1588.
Cited on Page 25.

[DJN+20] Ivan Damgård, Thomas Pelle Jakobsen, Jesper Buus Nielsen, Jakob Illeborg Pagter, and Michael Bæksvang Østergaard. Fast threshold ECDSA with honest majority. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 20: 12th International Conference on Security in Communication Networks*, volume 12238 of *Lecture Notes in Computer Science*, pages 382–400, Amalfi, Italy, September 14–16, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-57990-6_19.
Cited on Page 135.

[dK22] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis

and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022, Part II*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-15979-4_11.
Cited on Page 24.

[DKLs19] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy*, pages 1051–1066, San Francisco, CA, USA, May 19–23, 2019. IEEE Computer Society Press. doi:10.1109/SP.2019.00024.
Cited on Page 135.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in ad hoc groups. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_36.
Cited on Page 4.

[DKXY03] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 130–144, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany. doi:10.1007/3-540-36288-6_10.
Cited on Page 4.

[DOK+20] Anders P. K. Dalskov, Claudio Orlandi, Marcel Keller, Kris Shrishak, and Haya Shulman. Securing DNSSEC keys via threshold ECDSA from generic MPC. In Liqun Chen, Ninghui Li, Kaitai Liang, and Steve A. Schneider, editors, *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part II*, volume 12309 of *Lecture Notes in Computer Science*, pages 654–673, Guildford, UK, September 14–18, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-59013-0_32.
Cited on Page 4, 133.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 99–130, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_5.
Cited on Page 83, 85, 88, 89, 94, 95, 135.

[DYX+22] Sourav Das, Thomas Yurek, Zhuolun Xiang, Andrew K. Miller, Lefteris Kokoris-Kogias, and Ling Ren. Practical asynchronous distributed key generation. In *2022 IEEE Symposium on Security and Privacy*, pages 2518–2534, San Francisco, CA, USA, May 22–26, 2022. IEEE Computer Society Press. doi:10.1109/SP46214.2022.9833584.
Cited on Page 135.

[EFH+21] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In Juan Garay, editor, *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography,*

Part I, volume 12710 of *Lecture Notes in Computer Science*, pages 451–480, Virtual Event, May 10–13, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-75245-3_17.
Cited on Page 4.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 129–147, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-40084-1_8.
Cited on Page 15, 155.

[FH21] Masayuki Fukumitsu and Shingo Hasegawa. A tightly secure ddh-based multisignature with public-key aggregation. *Int. J. Netw. Comput.*, 11(2):319–337, 2021. URL: http://www.ijnc.org/index.php/ijnc/article/view/257.
Cited on Page 83, 85, 86, 88.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 233–253, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-48000-7_12.
Cited on Page 21, 24.

[FHSZ23] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, pages 386–400. ACM, 2023. doi:10.1145/3576915.3623219.
Cited on Page 85, 87.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. doi:10.1007/11535218_10.
Cited on Page 32.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. doi:10.1007/11818175_4.
Cited on Page 14, 24, 25.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96881-0_2.
Cited on Page 6, 16, 25, 133.

[FMY98] Yair Frankel, Philip D. MacKenzie, and Moti Yung. Robust efficient distributed RSA-key generation. In Brian A. Coan and Yehuda Afek, editors, *17th ACM Symposium Annual on Principles of Distributed Computing*, page 320, Puerto Vallarta, Mexico, June 28 – July 2, 1998. Association for Computing Machinery. doi:10.1145/277697.277779.
Cited on Page 135.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 63–95, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-45724-2_3.
Cited on Page 24.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. doi:10.1007/3-540-47721-7_12.
Cited on Page 63.

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-13190-5_10.
Cited on Page 24.

[FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1109–1123, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560655.
Cited on Page 85, 87, 135.

[FW22] Georg Fuchsbauer and Mathias Wolf. (Concurrently secure) blind schnorr from schnorr. Cryptology ePrint Archive, Report 2022/1676, 2022. https://eprint.iacr.org/2022/1676.
Cited on Page 163.

[GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 477–495, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-55220-5_27.
Cited on Page 24.

[GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1179–1194, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243859.
Cited on Page 135.

[GG20] Rosario Gennaro and Steven Goldfeder. One round threshold ECDSA with identifiable abort. Cryptology ePrint Archive, Report 2020/540, 2020. https://eprint.iacr.org/2020/540.
Cited on Page 135.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. doi:10.1109/SFCS.1984.715949.
Cited on Page 27, 30.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*, volume 9696 of *Lecture Notes in Computer Science*, pages 156–174, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-39555-5_9.
Cited on Page 135.

[Gha17] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017: 21st International Conference on Financial Cryptography and Data Security*, volume 10322 of *Lecture Notes in Computer Science*, pages 455–473, Sliema, Malta, April 3–7, 2017. Springer, Heidelberg, Germany.
Cited on Page 24.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 230–258, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_8.
Cited on Page 6.

[GHKR08] Rosario Gennaro, Shai Halevi, Hugo Krawczyk, and Tal Rabin. Threshold RSA for dynamic and ad-hoc groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 88–107, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-78967-3_6.
Cited on Page 135.

[GHKW16] Romain Gay, Dennis Hofheinz, Eike Kiltz, and Hoeteck Wee. Tightly CCA-secure encryption without pairings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 1–27, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-49890-3_1.
Cited on Page 6.

[GJ18] Kristian Gjøsteen and Tibor Jager. Practical and tightly-secure digital signatures and authenticated key exchange. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 95–125, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-96881-0_4.
Cited on Page 6, 164.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007. doi:10.1007/s00145-006-0347-3.
Cited on Page 134, 135.

[GJKW07] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the Diffie-Hellman problems. *Journal of Cryptology*, 20(4):493–514, October 2007. doi:10.1007/s00145-007-0549-3.
Cited on Page 89, 91, 134, 135, 136.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, MA, USA, October 11–14, 2003. IEEE Computer Society Press. doi:10.1109/SFCS.2003.1238185.
Cited on Page 6.

[GKS23] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice signatures from threshold homomorphic encryption. Cryptology ePrint Archive, Report 2023/1318, 2023. https://eprint.iacr.org/2023/1318.
Cited on Page 135.

[GKSŚ20] Adam Gągol, Jędrzej Kula, Damian Straszak, and Michał Świętek. Threshold ECDSA for decentralized asset custody. Cryptology ePrint Archive, Report 2020/498, 2020. https://eprint.iacr.org/2020/498.
Cited on Page 135.

[GMM+22] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 1259–1273, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560637.
Cited on Page 4.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.
Cited on Page 3, 5.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.
Cited on Page 6.

[Goo] Google. Vpn by google one, explained. https://one.google.com/about/vpn/howitworks. Accessed: 2023-12-22.
Cited on Page 3.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. doi:10.1007/11818175_6.
Cited on Page 84.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008. URL: https://doi.org/10.1016/j.dam.2007.12.010, doi:10.1016/J.DAM.2007.12.010.
Cited on Page 17.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press. doi:10.1145/1374376.1374407.
Cited on Page 3.

[GPZZ19] Panagiotis Grontas, Aris Pagourtzis, Alexandros Zacharakis, and Bingsheng Zhang. Towards everlasting privacy and efficient coercion resistance in remote electronic voting. In Aviv Zohar, Ittay Eyal, Vanessa Teague, Jeremy Clark, Andrea Bracciali, Federico Pintore, and Massimiliano Sala, editors, *FC 2018 Workshops*, volume 10958 of *Lecture Notes in Computer Science*, pages 210–231, Nieuwpoort, Curaçao, March 2, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-58820-8_15.
Cited on Page 3, 21.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, May 25–27, 1988. Springer, Heidelberg, Germany. doi:10.1007/3-540-45961-8_11.
Cited on Page 84.

[GQ90] Louis C. Guillou and Jean-Jacques Quisquater. A "paradoxical" indentity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *Advances in Cryptology – CRYPTO'88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231, Santa Barbara, CA, USA, August 21–25, 1990. Springer, Heidelberg, Germany. doi:10.1007/0-387-34799-2_16.
Cited on Page 32.

[Gro09] Jens Groth. Homomorphic trapdoor commitments to group elements. Cryptology ePrint Archive, Report 2009/007, 2009. https://eprint.iacr.org/2009/007.
Cited on Page 84, 90.

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 630–648, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-22792-9_36.
Cited on Page 24.

[GRSB19] Sharon Goldberg, Leonid Reyzin, Omar Sagga, and Foteini Baldimtsi. Efficient noninteractive certification of RSA moduli and beyond. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 700–727, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-34618-8_24.
Cited on Page 22, 63.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-78967-3_24.
Cited on Page 84, 90.

[GS23] Jens Groth and Victor Shoup. Fast batched asynchronous distributed key generation. Cryptology ePrint Archive, Report 2023/1175, 2023. https://eprint.iacr.org/2023/1175.
Cited on Page 135.

[HBG16] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *Lecture Notes in Computer Science*, pages 43–60, Christ Church, Barbados, February 26, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53357-4_4.
Cited on Page 3, 21, 24.

[HJ12] Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 590–607, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-32009-5_35.
Cited on Page 6.

[HJK+21] Shuai Han, Tibor Jager, Eike Kiltz, Shengli Liu, Jiaxin Pan, Doreen Riepel, and Sven Schäge. Authenticated key exchange and signatures with tight security in the standard model. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 670–700, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84259-8_23.
Cited on Page 6, 164.

[HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 345–375, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-17659-4_12.
Cited on Page 7, 21, 22, 25, 49, 51, 53, 59, 60, 97, 134, 140, 163.

[HKLN20] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 500–529, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-56880-1_18.
Cited on Page 21.

[HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. Universal signature aggregators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 3–34, Sofia,

Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-46803-6_1.
Cited on Page 4, 87.

[HLTW24] Lucjan Hanzlik, Julian Loss, Sri AravindaKrishnan Thyagarajan, and Benedikt Wagner. Sweep-UC: Swapping coins privately. In *2024 IEEE Symposium on Security and Privacy*, San Francisco, CA, USA, May 2024. IEEE Computer Society Press. doi:10.1109/SP54263.2024.00081.
Cited on Page vii, 3, 4.

[HLW22] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Rai-choo! Evolving blind signatures to the next level. Cryptology ePrint Archive, Report 2022/1350, 2022. https://eprint.iacr.org/2022/1350.
Cited on Page 20.

[HLW23a] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Rai-choo! Evolving blind signatures to the next level. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 753–783, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30589-4_26.
Cited on Page vii, 6, 9, 20, 25, 28, 64.

[HLW23b] Lucjan Hanzlik, Julian Loss, and Benedikt Wagner. Token meets wallet: Formalizing privacy and revocation for FIDO2. In *2023 IEEE Symposium on Security and Privacy*, pages 1491–1508, San Francisco, CA, USA, May 21–25, 2023. IEEE Computer Society Press. doi:10.1109/SP46215.2023.10179373.
Cited on Page vii.

[Hof17] Dennis Hofheinz. Adaptive partitioning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 489–518, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-56617-7_17.
Cited on Page 6.

[HW18] Susan Hohenberger and Brent Waters. Synchronized aggregate signatures from the RSA assumption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 197–229, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-78375-8_7.
Cited on Page 85, 87.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-45146-4_9.
Cited on Page 24, 34.

[IN83] Kazuharu Itakura and Katsuhiro Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983.
Cited on Page 4, 83, 85, 135.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 332–354, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-44647-8_20.
Cited on Page 4.

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 221–242, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_16.
Cited on Page 134, 135.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany. doi:10.1007/BFb0052233.
Cited on Page 3, 21, 24.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-81652-0_2.
Cited on Page 134, 159, 163.

[KGS23] Chelsea Komlo, Ian Goldberg, and Douglas Stebila. A formal treatment of distributed key generation, and new constructions. Cryptology ePrint Archive, Report 2023/292, 2023. https://eprint.iacr.org/2023/292.
Cited on Page 135.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, second edition, 2014.
Cited on Page 13, 14, 15.

[KLLQ23] Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI -otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 729–761, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38548-3_24.
Cited on Page 25.

[KLP17] Eike Kiltz, Julian Loss, and Jiaxin Pan. Tightly-secure signatures from five-move identification protocols. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 68–94, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-70700-6_3.
Cited on Page 134, 135, 136, 142.

[KLR21] Jonathan Katz, Julian Loss, and Michael Rosenberg. Boosting the security of blind signature schemes. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 468–492, Singapore, December 6–10, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-92068-5_16.
Cited on Page 7, 21, 22, 23, 25, 26, 28, 29, 48, 50, 51, 97, 134, 140.

[KLR23] Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ros: Implication for isogeny and lattice-based blind signatures. Cryptology ePrint Archive, Report 2023/1603, 2023. https://eprint.iacr.org/2023/1603.
Cited on Page 25.

[KLX22] Julia Kastner, Julian Loss, and Jiayu Xu. On pairing-free blind signature schemes in the algebraic group model. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 468–497, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-97131-1_16.
Cited on Page 5, 24.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. Optimal security proofs for signatures from identification schemes. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 33–61, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany. doi:10.1007/978-3-662-53008-5_2.
Cited on Page 6, 8, 84, 85, 88, 91, 109, 127, 135, 136, 164.

[KMS20] Eleftherios Kokoris-Kogias, Dahlia Malkhi, and Alexander Spiegelman. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1751–1767, Virtual Event, USA, November 9–13, 2020. ACM Press. doi:10.1145/3372297.3423364.
Cited on Page 135.

[KNR23] Julia Kastner, Ky Nguyen, and Michael Reichle. Pairing-free blind signatures from standard assumptions in the rom. Cryptology ePrint Archive, Report 2023/1810, 2023. https://eprint.iacr.org/2023/1810.
Cited on Page 25.

[KRS23] Shuichi Katsumata, Michael Reichle, and Yusuke Sakai. Practical round-optimal blind signatures in the ROM from standard assumptions. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part II*, volume 14439 of *Lecture Notes in Computer Science*, pages 383–417, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-981-99-8724-5_12.
Cited on Page 25.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003: 10th Conference on Computer and Communications Security*, pages 155–164, Washington, DC, USA, October 27–30, 2003. ACM Press. doi:10.1145/948109.948

132.
Cited on Page 6, 8, 84, 85, 87, 88, 91, 109, 127, 136, 142, 164.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
Cited on Page 3.

[Lin22] Yehuda Lindell. Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Report 2022/374, 2022. https://eprint.iacr.org/2022/374.
Cited on Page 14, 134.

[LJY14] Benoît Libert, Marc Joye, and Moti Yung. Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM Symposium Annual on Principles of Distributed Computing*, pages 303–312, Paris, France, July 15–18, 2014. Association for Computing Machinery. doi:10.1145/2611462.2611498.
Cited on Page 135.

[LLGW20] Xiangyu Liu, Shengli Liu, Dawu Gu, and Jian Weng. Two-pass authenticated key exchange with explicit authentication and tight security. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 785–814, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64834-3_27.
Cited on Page 6.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 74–90, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. doi:10.1007/978-3-540-24676-3_5.
Cited on Page 4, 87.

[LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1837–1854, Toronto, ON, Canada, October 15–19, 2018. ACM Press. doi:10.1145/3243734.3243788.
Cited on Page 4, 133, 135.

[LOS+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. doi:10.1007/11761679_28.
Cited on Page 4, 83, 85, 87.

[LP01] Anna Lysyanskaya and Chris Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 331–350, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-45682-1_20.
Cited on Page 134.

[LP20] Roman Langrehr and Jiaxin Pan. Unbounded HIBE with tight security. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 129–159, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-64834-3_5.
Cited on Page 6, 15.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-29011-4_43.
Cited on Page 3.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *Lecture Notes in Computer Science*, pages 1–12, Cirencester, UK, December 19–21, 2005. Springer, Heidelberg, Germany.
Cited on Page 6.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001: 8th Conference on Computer and Communications Security*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press. doi:10.1145/501983.502017.
Cited on Page 83.

[MOT+11] Prateek Mittal, Femi G. Olumofin, Carmela Troncoso, Nikita Borisov, and Ian Goldberg. PIR-tor: Scalable anonymous communication using private information retrieval. In *USENIX Security 2011: 20th USENIX Security Symposium*, San Francisco, CA, USA, August 8–12, 2011. USENIX Association.
Cited on Page 4.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple schnorr multi-signatures with applications to bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019. doi:10.1007/s10623-019-00608-x.
Cited on Page 83, 85, 86, 88, 135, 163.

[NRS20] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint.iacr.org/2020/1261.
Cited on Page 96.

[NRS21] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology – CRYPTO 2021, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 189–221, Virtual Event, August 16–20, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-84242-0_8.
Cited on Page 16, 17, 83, 84, 85, 86, 96, 135, 163.

[NRSW20] Jonas Nick, Tim Ruffing, Yannick Seurin, and Pieter Wuille. MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020: 27th Conference on Computer and Communications Security*, pages 1717–1731, Virtual Event, USA, November 9–13, 2020.

ACM Press. doi:10.1145/3372297.3417236.
Cited on Page 83, 85, 86, 135, 163.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press. doi:10.1145/73007.73011.
Cited on Page 3.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. doi:10.1007/3-540-48071-4_3.
Cited on Page 7, 21, 22, 28, 48, 163.

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany. doi:10.1007/11681878_5.
Cited on Page 24.

[OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 324–337, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. doi:10.1007/3-540-46766-1_27.
Cited on Page 3, 21.

[OO93] Kazuo Ohta and Tatsuaki Okamoto. A digital multisignature scheme based on the Fiat-Shamir scheme. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'91*, volume 739 of *Lecture Notes in Computer Science*, pages 139–148, Fujiyoshida, Japan, November 11–14, 1993. Springer, Heidelberg, Germany. doi:10.1007/3-540-57332-1_11.
Cited on Page 85.

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 109–118, San Jose, CA, USA, June 6–8, 2011. ACM Press. doi:10.1145/1993636.1993652.
Cited on Page 24.

[Ped91] Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 522–526, Brighton, UK, April 8–11, 1991. Springer, Heidelberg, Germany. doi:10.1007/3-540-46416-6_47.
Cited on Page 4, 87, 133.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. doi:10.1007/3-540-46766-1_9.
Cited on Page 32, 84, 135.

[Poi98] David Pointcheval. Strengthened security for blind signatures. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 391–405, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany. doi:10.1007/BFb0054141.
Cited on Page 21.

[Pol78] John M Pollard. Monte carlo methods for index computation mod p. *Mathematics of computation*, 32(143):918–924, 1978.
Cited on Page 48.

[PS96] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265, Kyongju, Korea, November 3–7, 1996. Springer, Heidelberg, Germany. doi:10.1007/BFb0034852.
Cited on Page 7.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. doi:10.1007/s001450010003.
Cited on Page 3, 21, 163.

[PW21] Jiaxin Pan and Benedikt Wagner. Short identity-based signatures with tight security from lattices. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 360–379, Daejeon, South Korea, July 20–22, 2021. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-81293-5_19.
Cited on Page 4.

[PW22] Jiaxin Pan and Benedikt Wagner. Lattice-based signatures with tight adaptive corruptions and more. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 13178 of *Lecture Notes in Computer Science*, pages 347–378, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-97131-1_12.
Cited on Page 6, 164.

[PW23a] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 597–627, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30589-4_21.
Cited on Page vii, 6, 9, 82, 84, 87, 92, 96, 134.

[PW23b] Jiaxin Pan and Benedikt Wagner. Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. Cryptology ePrint Archive, Report 2023/198, 2023. https://eprint.iacr.org/2023/198.
Cited on Page 82.

[PW23c] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. Cryptology ePrint Archive, Paper 2023/1613, 2023. https://eprint.iacr.org/2023/1613. URL: https://eprint.iacr.org/2023/1613.
Cited on Page 82.

[PW24] Jiaxin Pan and Benedikt Wagner. Toothpicks: More efficient fork-free two-round multi-signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 460–489, Zurich, Switherland, May 26–30, 2024. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-58716-0_16.
Cited on Page vii, 6, 9, 82, 87, 115.

[PWZ23a] Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Lattice-based authenticated key exchange with tight security. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 616–647, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-38554-4_20.
Cited on Page vii.

[PWZ23b] Jiaxin Pan, Benedikt Wagner, and Runzhi Zeng. Tighter security for generic authenticated key exchange in the QROM. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part IV*, volume 14441 of *Lecture Notes in Computer Science*, pages 401–433, Guangzhou, China, December 4–8, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-981-99-8730-6_13.
Cited on Page vii.

[Rab98] Tal Rabin. A simplified approach to threshold and proactive RSA. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 89–104, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. doi:10.1007/BFb0055722.
Cited on Page 135.

[RRJ+22] Tim Ruffing, Viktoria Ronge, Elliott Jin, Jonas Schneider-Bensch, and Dominique Schröder. ROAST: Robust asynchronous schnorr threshold signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022: 29th Conference on Computer and Communications Security*, pages 2551–2564, Los Angeles, CA, USA, November 7–11, 2022. ACM Press. doi:10.1145/3548606.3560583.
Cited on Page 134, 135, 163.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. doi:10.1145/359340.359342.
Cited on Page 18.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 552–565, Gold Coast, Australia, December 9–13, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-45682-1_32.
Cited on Page 4.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991. doi:10.1007/BF00196725.
Cited on Page 3, 88, 134, 135, 163, 164.

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01: 3rd International*

Conference on Information and Communication Security, volume 2229 of *Lecture Notes in Computer Science*, pages 1–12, Xian, China, November 13–16, 2001. Springer, Heidelberg, Germany.
Cited on Page 21.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.
Cited on Page 4.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. doi:10.1007/3-540-69053-0_18.
Cited on Page 6, 16, 25.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_15.
Cited on Page 135.

[Sho23] Victor Shoup. The many faces of schnorr. Cryptology ePrint Archive, Report 2023/1019, 2023. https://eprint.iacr.org/2023/1019.
Cited on Page 135.

[SS01] Douglas R. Stinson and Reto Strobl. Provably secure distributed Schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *ACISP 01: 6th Australasian Conference on Information Security and Privacy*, volume 2119 of *Lecture Notes in Computer Science*, pages 417–434, Sydney, NSW, Australia, July 2–4, 2001. Springer, Heidelberg, Germany. doi:10.1007/3-540-47719-5_33.
Cited on Page 134.

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on Theory of Computing*, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. doi:10.1145/2591796.2591825.
Cited on Page 27, 29.

[TMM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^{2}L: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy*, pages 1834–1851, San Francisco, CA, USA, May 24–27, 2021. IEEE Computer Society Press. doi:10.1109/SP40001.2021.00111.
Cited on Page 4.

[TSS+23] Kaoru Takemure, Yusuke Sakai, Bagus Santoso, Goichiro Hanaoka, and Kazuo Ohta. More efficient two-round multi-signature scheme with provably secure parameters. Cryptology ePrint Archive, Report 2023/155, 2023. https://eprint.iacr.org/2023/155.
Cited on Page 84, 85, 86, 87.

[TZ22] Stefano Tessaro and Chenzhi Zhu. Short pairing-free blind signatures with exponential security. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 782–811, Trondheim, Norway, May 30 – June 3, 2022. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-07085-3_27.
Cited on Page 24, 133.

[TZ23] Stefano Tessaro and Chenzhi Zhu. Threshold and multi-signature schemes from linear hash functions. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 628–658, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany. doi:10.1007/978-3-031-30589-4_22.
Cited on Page 85, 86, 87, 133, 134, 135, 138, 140, 156, 159.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. doi:10.1007/3-540-45708-9_19.
Cited on Page 21.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. doi:10.1007/11426639_7.
Cited on Page 3.

[WHL22] Benedikt Wagner, Lucjan Hanzlik, and Julian Loss. PI-cut-choo! Parallel instance cut and choose for practical blind signatures. Cryptology ePrint Archive, Report 2022/007, 2022. https://eprint.iacr.org/archive/2022/007/20220107:165256.
Cited on Page 20, 27.

A

Additional Pseudocode

Pseudocode for Section 4.4

Alg $\mathsf{Setup}(1^\lambda)$
01 **return** $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$

Alg $\mathsf{Gen}(\mathsf{par})$
02 $\mathsf{sk} := x \xleftarrow{\$} \mathcal{D},\ \mathsf{pk} := X := \mathsf{F}(x)$
03 **return** $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Agg}(\mathcal{P})$
04 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
05 **for** $i \in [N]$: **parse** $X_i := \mathsf{pk}_i$
06 **for** $i \in [N]$: $a_i := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_i)$
07 **return** $\tilde{\mathsf{pk}} := \tilde{X} := \sum_{i=1}^{N} a_i \cdot X_i$

Alg $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$
08 **parse** $\tilde{X} := \tilde{\mathsf{pk}},\ (\mathsf{com}, s, \varphi) := \sigma$
09 $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$
10 $R := \mathsf{F}(s) - c \cdot \tilde{X}$
11 $\mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
12 **if** $\mathsf{com} \neq \mathsf{Com}(\mathsf{ck}, R; \varphi)$: **return** 0
13 **return** 1

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$
14 $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$
15 **return** $\mathsf{VerAgg}(\tilde{\mathsf{pk}}, \mathsf{m}, \sigma)$

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$
16 **parse** $x_1 := \mathsf{sk}_1$
17 $\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}),\ \mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
18 $r_1 \xleftarrow{\$} \mathcal{D},\ R_1 := \mathsf{F}(r_1),\ \varphi_1 \xleftarrow{\$} \mathcal{G}$
19 $\mathsf{pm}_{1,1} := \mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$
20 $St_1 := (\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1, \mathsf{m})$
21 **return** $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
22 **parse** $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
23 **parse** $(\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1, \mathsf{m}) := St_1$
24 **for** $i \in [N]$: **parse** $\mathsf{com}_i := \mathsf{pm}_{1,i}$
25 $\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$
26 $c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$
27 $a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$
28 $s_1 := c \cdot a_1 \cdot x_1 + r_1$
29 $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$
30 **return** $(\mathsf{pm}_{2,1}, St_2 := \mathsf{com})$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
31 **parse** $\mathsf{com} := St_2 = \mathsf{com}$
32 **parse** $(\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N}) := \mathcal{M}_2$
33 **for** $i \in [N]$: **parse** $(s_i, \varphi_i) := \mathsf{pm}_{2,i}$
34 $s := \sum_{i=1}^{N} s_i,\ \varphi := \bigoplus_{i=1}^{N} \varphi_i$
35 **return** $\sigma := (\mathsf{com}, s, \varphi)$

Figure A.1: The multi-signature scheme ChopsKA[LF, CMT] = (Setup, Gen, Sig, Ver) with key aggregation for a linear function family LF = (LF.Gen, F) and a weakly equivocable commitment scheme CMT = (BGen, TGen, Com, TCom, TCol). Scheme ToothKA[LF, CMT] (Section 4.5.2) is the same, but CMT is assumed to be a weakly equivocable coset commitment scheme.

Game $\mathbf{G}_0$-$\mathbf{G}_8$

$\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$
$\mathsf{sk} := x_1 \xleftarrow{\$} \mathcal{D},\ \mathsf{pk} := X_1 := \mathsf{F}(x)$ / $\mathbf{G}_0$-$\mathbf{G}_5$
$\mathsf{pk}_1 := X_1 \xleftarrow{\$} \mathcal{R}$ / $\mathbf{G}_6$-$\mathbf{G}_8$
if $(\mathsf{par}, x_1) \notin \mathsf{Good}$: **abort** / $\mathbf{G}_1$-$\mathbf{G}_4$
$(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathrm{SIG}_0, \mathrm{SIG}_1, \mathsf{H}, \mathsf{H}_a, \mathsf{H}_c}(\mathsf{par}, \mathsf{pk}_1)$
if $\mathsf{pk} \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}^*)$ / $\mathbf{G}_2$-$\mathbf{G}_8$
if $b[\tilde{\mathsf{pk}}, \mathsf{m}^*] = 0$: **return** 0 / $\mathbf{G}_2$-$\mathbf{G}_8$
parse $\tilde{X} := \tilde{\mathsf{pk}},\ (\mathsf{com}^*, s^*, \varphi^*) := \sigma^*$ / $\mathbf{G}_7$-$\mathbf{G}_8$
$c^* := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*),\ R^* := \mathsf{F}(s^*) - c^* \cdot \tilde{X}$ / $\mathbf{G}_7$-$\mathbf{G}_8$
if $R^* \neq r[\tilde{\mathsf{pk}}, \mathsf{com}^*, \mathsf{m}^*]$: **return** 0 / $\mathbf{G}_7$-$\mathbf{G}_8$
return $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Oracle $\mathrm{SIG}_0(\mathcal{P}, \mathsf{m})$

parse $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
if $\mathsf{pk}_1 \neq \mathsf{pk}$: **return** $\bot$
$\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\},\ ctr := ctr + 1,\ sid := ctr,\ \mathsf{round}[sid] := 1$
$\tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P}),\ \mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$
if $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$: **abort** / $\mathbf{G}_2$-$\mathbf{G}_8$
$r_1 \xleftarrow{\$} \mathcal{D},\ R_1 := \mathsf{F}(r_1),\ \varphi_1 \xleftarrow{\$} \mathcal{G}$ / $\mathbf{G}_0$-$\mathbf{G}_3$
$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$ / $\mathbf{G}_0$-$\mathbf{G}_3$
$St_1 := (\tilde{\mathsf{pk}}, x_1, r_1, \varphi_1)$ / $\mathbf{G}_0$-$\mathbf{G}_3$
$(\mathsf{com}_1, St) \leftarrow \mathsf{TCom}(\mathsf{ck}, tr[\tilde{\mathsf{pk}}, \mathsf{m}])$ / $\mathbf{G}_4$-$\mathbf{G}_8$
$St_1 := St$ / $\mathbf{G}_4$-$\mathbf{G}_8$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_{1,1} := \mathsf{com}_1, St_1)$
return $(\mathsf{pm}_1[sid], sid)$

Oracle $\mathrm{SIG}_1(sid, \mathcal{M}_1)$

if $\mathsf{round}[sid] \neq 1$: **return** $\bot$
parse $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
if $\mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1}$: **return** $\bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1$
parse $(x_1, r_1, \varphi_1) := St_1$ / $\mathbf{G}_0$-$\mathbf{G}_3$
parse $St := St_1$ / $\mathbf{G}_4$-$\mathbf{G}_8$
for $i \in [N]$: **parse** $\mathsf{com}_i := \mathsf{pm}_{1,i}$
$\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i,\ c := \mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}),\ a_1 := \mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk}_1)$
$s_1 := c \cdot a_1 \cdot x_1 + r_1$ / $\mathbf{G}_0$-$\mathbf{G}_3$
$(\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St, c \cdot a_1)$ / $\mathbf{G}_4$-$\mathbf{G}_8$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_{2,1} := (s_1, \varphi_1), St_2 := \mathsf{com})$
return $\mathsf{pm}_2[sid]$

Figure A.2: The games $\mathbf{G}_0$-$\mathbf{G}_8$ used in the proof of Theorem 4.1. Lines with highlighted comments are only executed in the respective games. The random oracles are defined in Figure A.3.

Oracle $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ / $\mathbf{G}_0$-$\mathbf{G}_2$

01 **if** $h[\tilde{\mathsf{pk}}, \mathsf{m}] = \bot$:
02 $\quad b[\tilde{\mathsf{pk}}, \mathsf{m}] \leftarrow \mathcal{B}_\gamma$ / $\mathbf{G}_2$
03 $\quad h[\tilde{\mathsf{pk}}, \mathsf{m}] \xleftarrow{\$} \mathcal{K}$
04 **return** $h[\tilde{\mathsf{pk}}, \mathsf{m}]$

Oracle $\mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ / $\mathbf{G}_3$-$\mathbf{G}_8$

05 **if** $h[\tilde{\mathsf{pk}}, \mathsf{m}] = \bot$:
06 $\quad b[\tilde{\mathsf{pk}}, \mathsf{m}] \leftarrow \mathcal{B}_\gamma$
07 $\quad$ **if** $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 0$:
08 $\quad\quad (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_1)$
09 $\quad\quad tr[\tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{td}$
10 $\quad$ **if** $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$:
11 $\quad\quad \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$
12 $\quad h[\tilde{\mathsf{pk}}, \mathsf{m}] := \mathsf{ck}$
13 **return** $h[\tilde{\mathsf{pk}}, \mathsf{m}]$

Oracle $\mathsf{H}_c(\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m})$

14 **if** $h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] = \bot$:
15 $\quad \mathsf{ck} := \mathsf{H}(\tilde{\mathsf{pk}}, \mathsf{m})$ / $\mathbf{G}_7$-$\mathbf{G}_8$
16 $\quad$ **if** $b[\tilde{\mathsf{pk}}, \mathsf{m}] = 1$: / $\mathbf{G}_7$-$\mathbf{G}_8$
17 $\quad\quad R \leftarrow \mathsf{Ext}(\mathsf{ck}, \mathsf{com})$ / $\mathbf{G}_7$-$\mathbf{G}_8$
18 $\quad\quad r[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] := R$ / $\mathbf{G}_7$-$\mathbf{G}_8$
19 $\quad h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] \xleftarrow{\$} \mathcal{S}$
20 **return** $h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}]$

Oracle $\mathsf{H}_a(\langle \mathcal{P} \rangle, \mathsf{pk})$

21 **if** $h_a[\langle \mathcal{P} \rangle, \mathsf{pk}] = \bot$:
22 $\quad h_a[\langle \mathcal{P} \rangle, \mathsf{pk}] \xleftarrow{\$} \mathcal{S}$
23 **if** $\mathsf{pk} = \mathsf{pk}_1$:
24 $\quad\quad \tilde{\mathsf{pk}} := \mathsf{Agg}(\mathcal{P})$ / $\mathbf{G}_8$
25 $\quad\quad$ **if** $\exists (\mathsf{com}, \mathsf{m})$ s.t. / $\mathbf{G}_8$
$\quad\quad h_c[\tilde{\mathsf{pk}}, \mathsf{com}, \mathsf{m}] \neq \bot$: **abort** / $\mathbf{G}_8$
26 **return** $h_a[\langle \mathcal{P} \rangle, \mathsf{pk}]$

Figure A.3: The random oracles that are used in the proof of Theorem 4.1. Lines with highlighted comments are only executed in the respective games. Algorithm Ext is the (unbounded) extractor for the statistical binding property of CMT.

Alg $\mathsf{Setup}(1^\lambda)$
01 **return** $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$

Alg $\mathsf{Gen}(\mathsf{par})$
02 $x_0, x_1 \xleftarrow{\$} \mathcal{D}, \mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
03 $X_0 := \mathsf{F}(x_0), X_1 := \mathsf{F}(x_1)$
04 $\mathsf{pk} := (X_0, X_1), \mathsf{sk} := (x_0, x_1, \mathsf{seed})$
05 **return** $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$
06 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
07 **parse** $(\sigma_0, \sigma_1, B) := \sigma$
08 **parse** $(\mathsf{com}_0, s_0, \varphi_0) := \sigma_0$
09 **parse** $(\mathsf{com}_1, s_1, \varphi_1) := \sigma_1$
10 **parse** $b_1 \ldots b_N := B \in \{0,1\}^N$
11 **for** $i \in [N]$:
12 **parse** $(X_{i,0}, X_{i,1}) := \mathsf{pk}_i$
13 $c_{i,0} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0)$
14 $c_{i,1} := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1)$
15 $R_0 := \mathsf{F}(s_0) - \sum_{i=1}^N c_{i,0} \cdot X_{i,b_i}$
16 $R_1 := \mathsf{F}(s_1) - \sum_{i=1}^N c_{i,1} \cdot X_{i,1-b_i}$
17 $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$
18 $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$
19 **if** $\mathsf{com}_0 \neq \mathsf{Com}(\mathsf{ck}_0, R_0; \varphi_0)$:
20 **return** 0
21 **if** $\mathsf{com}_1 \neq \mathsf{Com}(\mathsf{ck}_1, R_1; \varphi_1)$:
22 **return** 0
23 **return** 1

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$
24 **parse** $(x_{1,0}, x_{1,1}, \mathsf{seed}_1) := \mathsf{sk}_1$
25 $\mathsf{ck}_0 := \mathsf{H}(0, \langle\mathcal{P}\rangle, \mathsf{m})$
26 $\mathsf{ck}_1 := \mathsf{H}(1, \langle\mathcal{P}\rangle, \mathsf{m})$
27 $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
28 $r_{1,0}, r_{1,1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,0}, \varphi_{1,1} \xleftarrow{\$} \mathcal{G}$
29 $R_{1,0} := \mathsf{F}(r_{1,0}), R_{1,1} := \mathsf{F}(r_{1,1})$
30 $\mathsf{com}_{1,0} := \mathsf{Com}(\mathsf{ck}_0, R_{1,0}; \varphi_{1,0})$
31 $\mathsf{com}_{1,1} := \mathsf{Com}(\mathsf{ck}_1, R_{1,1}; \varphi_{1,1})$
32 $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1})$
33 $St_1 := (\mathsf{sk}_1, r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$
34 **return** $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
35 **parse** $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
36 **parse** $(\mathsf{sk}_1, r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1}) := St_1$
37 **for** $i \in [N]$:
38 **parse** $(b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1}) := \mathsf{pm}_{1,i}$
39 $B := b_1 \ldots b_N \in \{0,1\}^N$
40 $\mathsf{com}_0 := \bigotimes_{i\in[N]} \mathsf{com}_{i,0}$
41 $\mathsf{com}_1 := \bigotimes_{i\in[N]} \mathsf{com}_{i,1}$
42 $c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle\mathcal{P}\rangle, B, 0)$
43 $c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle\mathcal{P}\rangle, B, 1)$
44 $s_{1,0} := c_{1,0} \cdot x_{1,b_1} + r_{1,0}$
45 $s_{1,1} := c_{1,1} \cdot x_{1,1-b_1} + r_{1,1}$
46 $\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1})$
47 $St_2 := (\mathsf{com}_0, \mathsf{com}_1)$
48 **return** $(\mathsf{pm}_{2,1}, St_2)$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
49 **parse** $(\mathsf{com}_0, \mathsf{com}_1) := St_2$
50 **parse** $(\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N}) := \mathcal{M}_2$
51 **for** $i \in [N]$:
52 **parse** $(s_{i,0}, s_{i,1}, \varphi_{i,0}, \varphi_{i,1}) := \mathsf{pm}_{2,i}$
53 $s_0 := \sum_{i=1}^N s_{i,0},\ \varphi_0 := \bigoplus_{i=1}^N \varphi_{i,0}$
54 $s_1 := \sum_{i=1}^N s_{i,1},\ \varphi_1 := \bigoplus_{i=1}^N \varphi_{i,1}$
55 $\sigma_0 := (\mathsf{com}_0, \varphi_0, s_0)$
56 $\sigma_1 := (\mathsf{com}_1, \varphi_1, s_1)$
57 $\sigma := (\sigma_0, \sigma_1, B)$
58 **return** σ

Figure A.4: The multi-signature scheme $\mathsf{Chops}[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ for a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ and a weakly equivocable commitment scheme $\mathsf{CMT} = (\mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$.

Game $\mathbf{G}_0$-$\mathbf{G}_8$

01 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda),\ x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D},\ \mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda,\ X_{1,0} := \mathsf{F}(x_{1,0})$
02 **if** $(\mathsf{par}, x_{1,1}) \notin \mathsf{Good}$: **abort** / $\mathbf{G}_3$-$\mathbf{G}_5$
03 $X_{1,1} := \mathsf{F}(x_{1,1})$ / $\mathbf{G}_0$-$\mathbf{G}_6$
04 $X_{1,1} \xleftarrow{\$} \mathcal{R}$ / $\mathbf{G}_7$-$\mathbf{G}_8$
05 $\mathsf{pk}^* := (X_{1,0}, X_{1,1})$
06 $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_b,\mathsf{H}_c,\mathrm{SIG}_0,\mathrm{SIG}_1}(\mathsf{par}, \mathsf{pk}^*)$
07 **if** $\mathsf{pk}^* \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
08 **parse** $(\sigma_0^*, \sigma_1^*, B^*) := \sigma^*,\ b_1^* \dots b_N^* := B^* \in \{0,1\}^N$
09 **parse** $(\mathsf{com}_0^*, \varphi_0^*, s_0^*) := \sigma_0^*,\ (\mathsf{com}_1^*, \varphi_1^*, s_1^*) := \sigma_1^*$
10 **if** $\mathsf{bad} = 1$: **return** 0 / $\mathbf{G}_1$-$\mathbf{G}_8$
11 **if** $b_1^* \neq 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*)$: **return** 0 / $\mathbf{G}_2$-$\mathbf{G}_8$
12 **parse** $\{\mathsf{pk}_1 = \mathsf{pk}^*, \dots, \mathsf{pk}_N\} := \mathcal{P}^*$ / $\mathbf{G}_8$
13 **for** $i \in [N]$: / $\mathbf{G}_8$
14 **parse** $(X_{i,0}, X_{i,1}) := \mathsf{pk}_i$ / $\mathbf{G}_8$
15 $c_{i,0}^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 0)$ / $\mathbf{G}_8$
16 $c_{i,1}^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_1^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*, 1)$ / $\mathbf{G}_8$
17 $R_0^* := \mathsf{F}(s_0^*) - \sum_{i=1}^N c_{i,0}^* \cdot X_{i,b_i^*},\ R_1^* := \mathsf{F}(s_1^*) - \sum_{i=1}^N c_{i,1}^* \cdot X_{i,1-b_i^*}$ / $\mathbf{G}_8$
18 **if** $R_{1-b_1^*}^* \neq r[\mathsf{com}_{1-b_1^*}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*]$: **return** 0 / $\mathbf{G}_8$
19 **return** $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Figure A.5: The games $\mathbf{G}_0$-$\mathbf{G}_8$ used in the proof of Theorem 4.2. Lines with highlighted comments are only executed in the respective games. The signing oracles are defined in Figure A.6, and the random oracles are defined in Figure A.7.

Oracle $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$

$\mathbf{parse}\ \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
$\mathbf{if}\ \mathsf{pk}_1 \neq \mathsf{pk}^* : \mathbf{return}\ \bot$
$\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\},\ ctr := ctr + 1,\ sid := ctr,\ \mathsf{round}[sid] := 1$
$\mathsf{ck}_0 := \mathsf{H}(0, \langle \mathcal{P} \rangle, \mathsf{m}),\ \mathsf{ck}_1 := \mathsf{H}(1, \langle \mathcal{P} \rangle, \mathsf{m})$
$b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$
$r_{1,b_1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,b_1} \xleftarrow{\$} \mathcal{G},\ R_{1,b_1} := \mathsf{F}(r_{1,b_1})$
$\mathsf{com}_{1,b_1} := \mathsf{Com}(\mathsf{ck}_{b_1}, R_{1,b_1}; \varphi_{1,b_1})$
$r_{1,1-b_1} \xleftarrow{\$} \mathcal{D},\ \varphi_{1,1-b_1} \xleftarrow{\$} \mathcal{G},\ R_{1,1-b_1} := \mathsf{F}(r_{1,1-b_1})$ / $\mathbf{G}_0$-$\mathbf{G}_4$
$\mathsf{com}_{1,1-b_1} := \mathsf{Com}(\mathsf{ck}_{1-b_1}, R_{1,1-b_1}; \varphi_{1,1-b_1})$ / $\mathbf{G}_0$-$\mathbf{G}_4$
$St_1 := (r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1})$ / $\mathbf{G}_0$-$\mathbf{G}_4$
$(\mathsf{com}_{1,1-b_1}, St) \leftarrow \mathsf{TCom}(\mathsf{ck}_{1-b_1}, tr[\langle \mathcal{P} \rangle, \mathsf{m}])$ / $\mathbf{G}_5$-$\mathbf{G}_8$
$St_1 := (r_{1,b_1}, \varphi_{1,b_1}, St)$ / $\mathbf{G}_5$-$\mathbf{G}_8$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_{1,0}, \mathsf{com}_{1,1}), St_1)$
$\mathbf{return}\ (\mathsf{pm}_1[sid], sid)$

Oracle $\textsc{Sig}_1(sid, \mathcal{M}_1)$

$\mathbf{if}\ \mathsf{round}[sid] \neq 1 : \mathbf{return}\ \bot$
$\mathbf{parse}\ (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
$\mathbf{if}\ \mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1} : \mathbf{return}\ \bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1,\ St_1 := St_1[sid]$
$\mathbf{parse}\ (r_{1,0}, r_{1,1}, \varphi_{1,0}, \varphi_{1,1}) := St_1$ / $\mathbf{G}_0$-$\mathbf{G}_4$
$\mathbf{parse}\ (r_{1,b_1}, \varphi_{1,b_1}, St) := St_1$ / $\mathbf{G}_5$-$\mathbf{G}_8$
$\mathbf{for}\ i \in [N] : \mathbf{parse}\ (b_i, \mathsf{com}_{i,0}, \mathsf{com}_{i,1}) := \mathsf{pm}_{1,i}$
$B := b_1 \ldots b_N \in \{0,1\}^N$
$\mathsf{com}_0 := \bigotimes_{i \in [N]} \mathsf{com}_{i,0},\ \mathsf{com}_1 := \bigotimes_{i \in [N]} \mathsf{com}_{i,1}$
$c_{1,0} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_0, \mathsf{m}, \langle \mathcal{P} \rangle, B, 0),\ c_{1,1} := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}_1, \mathsf{m}, \langle \mathcal{P} \rangle, B, 1)$
$s_{1,b_1} := c_{1,b_1} \cdot x_{1,0} + r_{1,b_1}$
$s_{1,1-b_1} := c_{1,1-b_1} \cdot x_{1,1} + r_{1,1-b_1}$ / $\mathbf{G}_0$-$\mathbf{G}_4$
$(\varphi_{1,1-b_1}, R_{1-b_1}, s_{1,1-b_1}) \leftarrow \mathsf{TCol}(St, c_{1,1-b_1})$ / $\mathbf{G}_5$-$\mathbf{G}_8$
$St_2 := (\mathsf{com}_0, \mathsf{com}_1)$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_{2,1} := (s_{1,0}, s_{1,1}, \varphi_{1,0}, \varphi_{1,1}), St_2)$
$\mathbf{return}\ \mathsf{pm}_2[sid]$

Figure A.6: The signing oracles that are used in the proof of Theorem 4.2. Lines with highlighted comments are only executed in the respective games.

Oracle $\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ / $\mathbf{G}_0$-$\mathbf{G}_3$

01 **if** $h[b, \langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
02 $\quad h[b, \langle\mathcal{P}\rangle, \mathsf{m}] \xleftarrow{\$} \mathcal{K}$
03 **return** $h[b, \langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m})$ / $\mathbf{G}_4$-$\mathbf{G}_8$

04 **if** $h[b, \langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
05 $\quad$ **if** $b = 1 - \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$:
06 $\quad\quad (\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1})$
07 $\quad\quad tr[\langle\mathcal{P}\rangle, \mathsf{m}] := \mathsf{td}$
08 $\quad$ **if** $b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$:
09 $\quad\quad \mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$
10 $\quad h[b, \langle\mathcal{P}\rangle, \mathsf{m}] := \mathsf{ck}$
11 **return** $h[b, \langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\bar{\mathsf{H}}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

12 **if** $\bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
13 $\quad \bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}] \xleftarrow{\$} \{0,1\}$
14 **return** $\bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$ / $\mathbf{G}_0$-$\mathbf{G}_7$

15 **if** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b] = \bot$:
16 $\quad h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b] \xleftarrow{\$} \mathcal{S}$
17 **return** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b]$

Oracle $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b)$ / $\mathbf{G}_8$

18 **if** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b] = \bot$:
19 $\quad$ **if** $\mathsf{pk} = \mathsf{pk}^* \wedge b = \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$:
20 $\quad\quad R \leftarrow \mathsf{Ext}(\mathsf{H}(b, \langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$
21 $\quad\quad r[\mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] := R$
22 $\quad h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b] \xleftarrow{\$} \mathcal{S}$
23 **return** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B, b]$

Oracle $\mathsf{H}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

24 **if** $\mathsf{seed} = \mathsf{seed}_1$: $\mathsf{bad} := 1$ / $\mathbf{G}_1$-$\mathbf{G}_8$
25 **return** $\bar{\mathsf{H}}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

Figure A.7: The random oracles that are used in the proof of Theorem 4.2. Lines with highlighted comments are only executed in the respective games. Algorithm Ext is the (unbounded) extractor for the statistical binding property of CMT.

Pseudocode for Section 4.5

Alg $\mathsf{Setup}(1^\lambda)$
01 **return** $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$

Alg $\mathsf{Gen}(\mathsf{par})$
02 $x_0, x_1 \xleftarrow{\$} \mathcal{D}, \mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
03 $X_0 := \mathsf{F}(x_0), X_1 := \mathsf{F}(x_1)$
04 $\mathsf{pk} := (X_0, X_1), \mathsf{sk} := (x_0, x_1, \mathsf{seed})$
05 **return** $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Ver}(\mathcal{P}, \mathsf{m}, \sigma)$
06 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
07 **parse** $(\mathsf{com}, \varphi, s, B) := \sigma$
08 **parse** $b_1 \ldots b_N := B \in \{0,1\}^N$
09 **for** $i \in [N]$:
10 **parse** $(X_{i,0}, X_{i,1}) := \mathsf{pk}_i$
11 $c_i := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$
12 $R := \mathsf{F}(s) - \sum_{i=1}^N c_i \cdot X_{i,b_i}$
13 $\mathsf{ck} := \mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$
14 **if** $\mathsf{com} \neq \mathsf{Com}(\mathsf{ck}, R; \varphi)$: **return** 0
15 **return** 1

Alg $\mathsf{Sig}_0(\mathcal{P}, \mathsf{sk}_1, \mathsf{m})$
16 **parse** $(x_{1,0}, x_{1,1}, \mathsf{seed}_1) := \mathsf{sk}_1$
17 $\mathsf{ck} := \mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$
18 $b_1 := \mathsf{H}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
19 $r_1 \xleftarrow{\$} \mathcal{D},\ \varphi_1 \xleftarrow{\$} \mathcal{G},\ R_1 := \mathsf{F}(r_1)$
20 $\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$
21 $\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_1)$
22 $St_1 := (\mathsf{sk}_1, r_1, \varphi_1)$
23 **return** $(\mathsf{pm}_{1,1}, St_1)$

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
24 **parse** $(\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
25 **parse** $(\mathsf{sk}_1, r_1, \varphi_1) := St_1$
26 **for** $i \in [N]$:
27 **parse** $(b_i, \mathsf{com}_i) := \mathsf{pm}_{1,i}$
28 $B := b_1 \ldots b_N \in \{0,1\}^N$
29 $\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$
30 $c_1 := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$
31 $s_1 := c_1 \cdot x_{1,b_1} + r_1$
32 $\mathsf{pm}_{2,1} := (s_1, \varphi_1)$
33 $St_2 := \mathsf{com}$
34 **return** $(\mathsf{pm}_{2,1}, St_2)$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
35 **parse** $\mathsf{com} := St_2$
36 **parse** $(\mathsf{pm}_{2,1}, \ldots, \mathsf{pm}_{2,N}) := \mathcal{M}_2$
37 **for** $i \in [N]$: **parse** $(s_i, \varphi_i) := \mathsf{pm}_{2,i}$
38 $s := \sum_{i=1}^N s_i,\ \varphi := \bigoplus_{i=1}^N \varphi_i$
39 **return** $\sigma := (\mathsf{com}, \varphi, s, B)$

Figure A.8: The multi-signature scheme $\mathsf{Tooth}[\mathsf{LF}, \mathsf{CMT}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ for a linear function family $\mathsf{LF} = (\mathsf{LF.Gen}, \mathsf{F})$ and a weakly equivocable coset commitment scheme $\mathsf{CMT} = (\mathsf{TGen}, \mathsf{Com}, \mathsf{TCom}, \mathsf{TCol})$.

Game $\mathbf{G}_0$-$\mathbf{G}_8$

01 $b^* \xleftarrow{\$} \{0,1\}$ / $\mathbf{G}_2$-$\mathbf{G}_8$
02 $\mathsf{par} \leftarrow \mathsf{LF.Gen}(1^\lambda)$
03 $\mathsf{seed}_1 \xleftarrow{\$} \{0,1\}^\lambda$
04 $x_{1,0}, x_{1,1} \xleftarrow{\$} \mathcal{D},\ X_{1,0} := \mathsf{F}(x_{1,0}),\ X_{1,1} := \mathsf{F}(x_{1,1})$ / $\mathbf{G}_0$-$\mathbf{G}_6$
05 $x_{1,b^*} \xleftarrow{\$} \mathcal{D},\ X_{1,b^*} := \mathsf{F}(x_{1,b^*}),\ X_{1,1-b^*} \xleftarrow{\$} \mathcal{R}$ / $\mathbf{G}_7$-$\mathbf{G}_8$
06 **if** $(\mathsf{par}, x_{1,1-b^*}) \notin \mathsf{Good}$: **return** 0 / $\mathbf{G}_3$-$\mathbf{G}_5$
07 $\mathsf{pk}^* := (X_{1,0}, X_{1,1})$
08 $(\mathcal{P}^*, \mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_b,\mathsf{H}_c,\mathrm{SIG}_0,\mathrm{SIG}_1}(\mathsf{par}, \mathsf{pk}^*)$
09 **if** $\mathsf{pk}^* \notin \mathcal{P}^* \vee (\mathcal{P}^*, \mathsf{m}^*) \in \mathsf{Queried}$: **return** 0
10 **parse** $(\mathsf{com}^*, \varphi^*, s^*, B^*) := \sigma^*,\ \ b_1^* \ldots b_N^* := B^* \in \{0,1\}^N$
11 **if** $\mathsf{bad} = 1$: **return** 0 / $\mathbf{G}_1$-$\mathbf{G}_8$
12 **if** $b^* = b_1^* \vee \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P}^* \rangle, \mathsf{m}^*) = 1 - b^*$: **return** 0 / $\mathbf{G}_2$-$\mathbf{G}_8$
13 **parse** $\{\mathsf{pk}_1 = \mathsf{pk}^*, \ldots, \mathsf{pk}_N\} := \mathcal{P}^*$ / $\mathbf{G}_8$
14 **for** $i \in [N]$: / $\mathbf{G}_8$
15 **parse** $(X_{i,0}, X_{i,1}) := \mathsf{pk}_i$ / $\mathbf{G}_8$
16 $c_i^* := \mathsf{H}_c(\mathsf{pk}_i, \mathsf{com}_0^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*)$ / $\mathbf{G}_8$
17 $R^* := \mathsf{F}(s^*) - \sum_{i=1}^N c_i^* \cdot X_{i,b_i^*}$ / $\mathbf{G}_8$
18 **if** $R^* \notin r[\mathsf{com}^*, \mathsf{m}^*, \langle \mathcal{P}^* \rangle, B^*] + \mathsf{F}(\mathcal{D})$: **return** 0 / $\mathbf{G}_8$
19 **return** $\mathsf{Ver}(\mathcal{P}^*, \mathsf{m}^*, \sigma^*)$

Figure A.9: The games $\mathbf{G}_0$-$\mathbf{G}_8$ used in the proof of Theorem 4.5. Lines with highlighted comments are only executed in the respective games. The signing oracles are defined in Figure A.10, and the random oracles are defined in Figure A.11.

Oracle $\textsc{Sig}_0(\mathcal{P}, \mathsf{m})$

$\mathbf{parse}\ \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
$\mathbf{if}\ \mathsf{pk}_1 \neq \mathsf{pk}^* : \mathbf{return}\ \bot$
$\mathsf{Queried} := \mathsf{Queried} \cup \{(\mathcal{P}, \mathsf{m})\},\ ctr := ctr + 1,\ sid := ctr,\ \mathsf{round}[sid] := 1$
$\mathsf{ck} := \mathsf{H}(\langle \mathcal{P} \rangle, \mathsf{m})$
$b_1 := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle \mathcal{P} \rangle, \mathsf{m})$
$r_1 \xleftarrow{\$} \mathcal{D},\ \varphi_1 \xleftarrow{\$} \mathcal{G},\ R_1 := \mathsf{F}(r_{1,b_1})$
$\mathsf{com}_1 := \mathsf{Com}(\mathsf{ck}, R_1; \varphi_1)$
$St_1 := (r_1, \varphi_1)$
$\mathbf{if}\ b_1 = 1 - b^* :\ (\mathsf{com}_1, St_1) \leftarrow \mathsf{TCom}(\mathsf{ck}, tr[\langle \mathcal{P} \rangle, \mathsf{m}])$ / $\mathbf{G_5}$-$\mathbf{G_8}$
$(\mathsf{pm}_1[sid], St_1[sid]) := (\mathsf{pm}_{1,1} := (b_1, \mathsf{com}_1), St_1)$
$\mathbf{return}\ (\mathsf{pm}_1[sid], sid)$

Oracle $\textsc{Sig}_1(sid, \mathcal{M}_1)$

$\mathbf{if}\ \mathsf{round}[sid] \neq 1 : \mathbf{return}\ \bot$
$\mathbf{parse}\ (\mathsf{pm}_{1,1}, \ldots, \mathsf{pm}_{1,N}) := \mathcal{M}_1$
$\mathbf{if}\ \mathsf{pm}_1[sid] \neq \mathsf{pm}_{1,1} : \mathbf{return}\ \bot$
$\mathsf{round}[sid] := \mathsf{round}[sid] + 1,\ St_1 := St_1[sid]$
$\mathbf{parse}\ (r_1, \varphi_1) := St_1$ / $\mathbf{G_0}$-$\mathbf{G_4}$
$\mathbf{for}\ i \in [N] : \mathbf{parse}\ (b_i, \mathsf{com}_i) := \mathsf{pm}_{1,i}$
$B := b_1 \ldots b_N \in \{0,1\}^N$
$\mathsf{com} := \bigotimes_{i \in [N]} \mathsf{com}_i$
$c_1 := \mathsf{H}_c(\mathsf{pk}_1, \mathsf{com}, \mathsf{m}, \langle \mathcal{P} \rangle, B)$
$s_1 := c_1 \cdot x_{1,b_1} + r_1$ / $\mathbf{G_0}$-$\mathbf{G_4}$
$\mathbf{if}\ b_1 = 1 - b^* :\ (\varphi_1, R_1, s_1) \leftarrow \mathsf{TCol}(St_1, c_1)$ / $\mathbf{G_5}$-$\mathbf{G_8}$
$\mathbf{if}\ b_1 = b^* :\ s_1 := c_1 \cdot x_{1,b_1} + r_1$ / $\mathbf{G_5}$-$\mathbf{G_8}$
$St_2 := \mathsf{com}$
$(\mathsf{pm}_2[sid], St_2[sid]) := (\mathsf{pm}_{2,1} := (s_1, \varphi_1), St_2)$
$\mathbf{return}\ \mathsf{pm}_2[sid]$

Figure A.10: The signing oracles that are used in the proof of Theorem 4.5. Lines with highlighted comments are only executed in the respective games.

Oracle $\mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$ / $\mathbf{G}_0$-$\mathbf{G}_3$

01 **if** $h[\langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
02 $h[\langle\mathcal{P}\rangle, \mathsf{m}] \xleftarrow{\$} \mathcal{K}$
03 **return** $h[\langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m})$ / $\mathbf{G}_4$-$\mathbf{G}_8$

04 **if** $h[\langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
05 $b := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
06 **if** $b = 1 - b^*$:
07 $(\mathsf{ck}, \mathsf{td}) \leftarrow \mathsf{TGen}(\mathsf{par}, X_{1,1-b^*})$
08 $tr[\langle\mathcal{P}\rangle, \mathsf{m}] := \mathsf{td}$
09 **if** $b = b^*$:
10 $\mathsf{ck} \leftarrow \mathsf{BGen}(\mathsf{par})$
11 $h[\langle\mathcal{P}\rangle, \mathsf{m}] := \mathsf{ck}$
12 **return** $h[\langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\bar{\mathsf{H}}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

13 **if** $\bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}] = \bot$:
14 $\bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}] \xleftarrow{\$} \{0,1\}$
15 **return** $\bar{h}_b[\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m}]$

Oracle $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$ / $\mathbf{G}_0$-$\mathbf{G}_7$

16 **if** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] = \bot$:
17 $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] \xleftarrow{\$} \mathcal{S}$
18 **return** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B]$

Oracle $\mathsf{H}_c(\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B)$ / $\mathbf{G}_8$

19 **if** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] = \bot$:
20 **parse** $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\} := \mathcal{P}$
21 **parse** $b_1 \ldots b_N := B$
22 $b := \bar{\mathsf{H}}_b(\mathsf{seed}_1, \langle\mathcal{P}\rangle, \mathsf{m})$
23 **if** $\mathsf{pk} = \mathsf{pk}^* = \mathsf{pk}_1 \wedge b = b^*$:
24 $R \leftarrow \mathsf{Ext}(\mathsf{H}(\langle\mathcal{P}\rangle, \mathsf{m}), \mathsf{com})$
25 $r[\mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] := R$
26 $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B] \xleftarrow{\$} \mathcal{S}$
27 **return** $h_c[\mathsf{pk}, \mathsf{com}, \mathsf{m}, \langle\mathcal{P}\rangle, B]$

Oracle $\mathsf{H}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

28 **if** $\mathsf{seed} = \mathsf{seed}_1$: $\mathsf{bad} := 1$ / $\mathbf{G}_1$-$\mathbf{G}_8$
29 **return** $\bar{\mathsf{H}}_b(\mathsf{seed}, \langle\mathcal{P}\rangle, \mathsf{m})$

Figure A.11: The random oracles that are used in the proof of Theorem 4.5. Lines with highlighted comments are only executed in the respective games. Algorithm Ext is the (unbounded) extractor for the statistical coset binding property of CMT.

Pseudocode for Section 5.4

Alg $\mathsf{Setup}(1^\lambda)$
01 $\mathsf{par}' \leftarrow \mathsf{TLF.Gen}(1^\lambda),\ \ g \xleftarrow{\$} \mathcal{T}$
02 **return** $\mathsf{par} := (\mathsf{par}, g)$

Alg $\mathsf{Gen}(\mathsf{par})$
03 $a_0, \ldots, a_t \xleftarrow{\$} \mathcal{D}$
04 **for** $i \in [n]$: $\mathsf{sk}_i := \sum_{j=0}^{t} a_j i^j$
05 $\mathsf{pk} := \mathsf{pk}_0 := \mathsf{T}(g, a_0)$
06 **return** $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_n)$

Alg $\mathsf{Sig}_0(S, i, \mathsf{sk}_i, \mathsf{m})$
07 $h := \mathsf{H}(\mathsf{m})$
08 $r_i \xleftarrow{\$} \mathcal{D}$
09 $R_i^{(1)} := \mathsf{T}(g, r_i),\ R_i^{(2)} := \mathsf{T}(h, r_i)$
10 $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$
11 $\mathsf{com}_i := \hat{\mathsf{H}}(S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$
12 $\mathsf{pm}_1 := \mathsf{com}_i$
13 $St_1 := (i, S, \mathsf{sk}_i, h, \mathsf{m}, r_i)$
14 **return** (pm_1, St_1)

Alg $\mathsf{Sig}_1(St_1, \mathcal{M}_1)$
15 **parse** $(i, S, \mathsf{sk}_i, h, \mathsf{m}, r_i) := St_1$
16 $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$
17 $R_i^{(1)} := \mathsf{T}(g, r_i),\ R_i^{(2)} := \mathsf{T}(h, r_i)$
18 $\mathsf{pm}_2 := (R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$
19 **return** $(\mathsf{pm}_2, St_2 := (\mathcal{M}_1, St_1))$

Alg $\mathsf{Sig}_2(St_2, \mathcal{M}_2)$
20 **parse** $(\mathcal{M}_1, (i, S, \mathsf{sk}_i, h, \mathsf{m}, r_i)) := St_2$
21 **parse** $(\mathsf{com}_j)_{j \in S} := \mathcal{M}_1$
22 **parse** $((R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}))_{j \in S} := \mathcal{M}_2$
23 **for** $j \in S$:
24 **if** $\hat{\mathsf{H}}(S, j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) \neq \mathsf{com}_j$:
25 **return** $\perp$
26 $R^{(1)} := \sum_{j \in S} R_j^{(1)}$
27 $R^{(2)} := \sum_{j \in S} R_j^{(2)}$
28 $\mathsf{pk}^{(2)} := \sum_{j \in S} \ell_{j,S} \mathsf{pk}_j^{(2)}$
29 $c := \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$
30 **return** $\mathsf{pm}_3 := s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$

Alg $\mathsf{Combine}(S, \mathsf{m}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)$
31 **parse** $(\mathsf{com}_j)_{j \in S} := \mathcal{M}_1$
32 **parse** $((R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}))_{j \in S} := \mathcal{M}_2$
33 **parse** $(s_j)_{j \in S} := \mathcal{M}_3$
34 $R^{(1)} := \sum_{j \in S} R_j^{(1)}$
35 $R^{(2)} := \sum_{j \in S} R_j^{(2)}$
36 $\mathsf{pk}^{(2)} := \sum_{j \in S} \ell_{j,S} \mathsf{pk}_i^{(2)}$
37 $c := \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$
38 $s := \sum_{j \in S} s_j$
39 **return** $\sigma := (\mathsf{pk}^{(2)}, c, s)$

Alg $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma = (\mathsf{pk}^{(2)}, c, s))$
40 $h := \mathsf{H}(\mathsf{m}),\ \ R^{(1)} := \mathsf{T}(g, s) - c \cdot \mathsf{pk},\ \ R^{(2)} := \mathsf{T}(h, s) - c \cdot \mathsf{pk}^{(2)}$
41 **if** $c = \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$: **return** 1
42 **return** 0

Figure A.12: The (t, n)-threshold signature scheme $\mathsf{Twinkle}[\mathsf{TLF}] = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ for a tagged linear function family TLF.

Game $\mathbf{G}_0$-$\mathbf{G}_{11}$

01 $\mathsf{Pending} := \emptyset$ / $\mathbf{G}_7$-$\mathbf{G}_{11}$
02 $\mathsf{Reps} := \emptyset$ / $\mathbf{G}_9$-$\mathbf{G}_{11}$
03 $\mathsf{par}' \leftarrow \mathsf{TLF.Gen}(1^\lambda),\ \ g \xleftarrow{\$} \mathcal{T},\ \ \mathsf{par} := (\mathsf{par}', g)$
04 **if** $(\mathsf{par}', g) \notin \mathsf{Reg}$: **abort** / $\mathbf{G}_4$-$\mathbf{G}_{10}$
05 $a_0, \ldots, a_t \xleftarrow{\$} \mathcal{D}$
06 **for** $i \in [n]$: $\mathsf{sk}_i := \sum_{j=0}^{t} a_j i^j$
07 **for** $i \in [n]$: $\mathsf{pk}_i := \mathsf{T}(g, \mathsf{sk}_i)$
08 $\mathsf{pk} := \mathsf{pk}_0 := \mathsf{T}(g, a_0)$
09 $\textsc{Sig} := (\textsc{Next}, \textsc{Sig}_0, \textsc{Sig}_1, \textsc{Sig}_2)$
10 $(\mathsf{m}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sig},\textsc{Corr},\mathsf{H},\hat{\mathsf{H}},\bar{\mathsf{H}}}(\mathsf{par}, \mathsf{pk})$
11 **if** $b[\mathsf{m}^*] = 0$: **return** 0 / $\mathbf{G}_1$-$\mathbf{G}_{11}$
12 **if** $\exists x \neq x'$ s.t. $\hat{h}[x] = \hat{h}[x'] \neq \bot$: **return** 0 / $\mathbf{G}_6$-$\mathbf{G}_{11}$
13 **if** $\mathsf{m}^* \in \mathsf{Queried}$: **return** 0
14 **return** $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}^*, \sigma^*)$

Oracle $\textsc{Corr}(i)$

15 **if** $|\mathsf{Corrupted}| \geq t$: **return** $\bot$
16 $\mathsf{Corrupted} := \mathsf{Corrupted} \cup \{i\}$
17 **for** $sid \in \mathsf{Sessions}$ s.t. $\mathsf{round}[sid, i] = 1$: / $\mathbf{G}_5$-$\mathbf{G}_{11}$
18 **parse** $(i, S, h, \mathsf{m}, \mathsf{com}_i) := \mathsf{state}[sid, i]$ / $\mathbf{G}_5$-$\mathbf{G}_{11}$
19 $\mathsf{pk}_i^{(2)} := \mathsf{Translate}(tr[\mathsf{m}], \mathsf{pk}_i)$ / $\mathbf{G}_5$-$\mathbf{G}_{11}$
20 $r_i \xleftarrow{\$} \mathcal{D},\ R_i^{(1)} := \mathsf{T}(g, r_i),\ R_i^{(2)} := \mathsf{T}(h, r_i)$ / $\mathbf{G}_5$-$\mathbf{G}_{11}$
21 **if** $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] \neq \bot$: **abort** / $\mathbf{G}_5$-$\mathbf{G}_{11}$
22 $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] := \mathsf{com}_i$ / $\mathbf{G}_5$-$\mathbf{G}_{11}$
23 $\mathsf{state}[sid, i] := (i, S, h, \mathsf{m}, r_i)$ / $\mathbf{G}_5$-$\mathbf{G}_{11}$
24 **for** $sid \in \mathsf{Sessions}$ s.t. $\mathsf{round}[sid, i] \geq 2$: / $\mathbf{G}_{10}$-$\mathbf{G}_{11}$
25 **parse** $(\mathcal{M}_1, (i, S, h, \mathsf{m}, s_i, c)) := \mathsf{state}[sid, i]$ / $\mathbf{G}_{10}$-$\mathbf{G}_{11}$
26 $r_i := s_i - c \cdot \ell_{i,S} \cdot \mathsf{sk}_i$ / $\mathbf{G}_{10}$-$\mathbf{G}_{11}$
27 $\mathsf{state}[sid, i] := (\mathcal{M}_1, (i, S, h, \mathsf{m}, r_i))$ / $\mathbf{G}_{10}$-$\mathbf{G}_{11}$
28 $\mathsf{UpdatePending}()$ / $\mathbf{G}_7$-$\mathbf{G}_{11}$
29 **return** $(\mathsf{sk}_i, \mathsf{state}[\cdot, i])$

Figure A.13: Games $\mathbf{G}_0$-$\mathbf{G}_{11}$ in the proof of Theorem 5.1. Lines with highlighted are only executed in the respective games. Signing oracles are given in Figures A.14 to A.16. Random oracles are given in Figure A.17. Algorithm UpdatePending is given in Figure A.18. Oracle Next is as in Figure 5.2.

Oracle $\text{SIG}_0(sid, i)$

01 **if** $\mathsf{Allowed}(sid, i, 0, \bot) = 0$:
02 **return** $\bot$
03 $h := \mathsf{H}(\mathsf{m}),\ S := \mathsf{signers}[sid]$
04 **if** $b[\mathsf{m}] = 1$: **abort** / $\mathbf{G_1}$-$\mathbf{G_{11}}$
05 $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$ / $\mathbf{G_0}$-$\mathbf{G_2}$
06 $\mathsf{pk}_i^{(2)} := \mathsf{Translate}(tr[\mathsf{m}], \mathsf{pk}_i)$ / $\mathbf{G_3}$-$\mathbf{G_{11}}$
07 $r_i \xleftarrow{\$} \mathcal{D}$ / $\mathbf{G_0}$-$\mathbf{G_4}$
08 $R_i^{(1)} := \mathsf{T}(g, r_i),\ R_i^{(2)} := \mathsf{T}(h, r_i)$ / $\mathbf{G_0}$-$\mathbf{G_4}$
09 $\mathsf{com}_i := \hat{\mathsf{H}}(S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$ / $\mathbf{G_0}$-$\mathbf{G_4}$
10 $\mathsf{state}[sid, i] := (i, S, h, \mathsf{m}, r_i)$ / $\mathbf{G_0}$-$\mathbf{G_4}$
11 $\mathsf{com}_i \xleftarrow{\$} \{0,1\}^{2\lambda}$ / $\mathbf{G_5}$-$\mathbf{G_{11}}$
12 **if** $\exists (S', i') \neq (S, i)$ s.t. $(S', i', \mathsf{com}_i) \in \mathsf{Sim}$: **abort** / $\mathbf{G_5}$-$\mathbf{G_{11}}$
13 $\mathsf{Sim} := \mathsf{Sim} \cup \{(S, i, \mathsf{com}_i)\}$ / $\mathbf{G_5}$-$\mathbf{G_{11}}$
14 $\mathsf{state}[sid, i] := (i, S, h, \mathsf{m}, \mathsf{com}_i)$ / $\mathbf{G_5}$-$\mathbf{G_{11}}$
15 $\mathsf{round}[sid, i] := 1$
16 **return** $\mathsf{pm}_1[sid, i] := \mathsf{com}_i$

Figure A.14: Signing Oracle SIG_0 in the proof of Theorem 5.1. Lines with highlighted are only executed in the respective games. Algorithm Allowed is as in Figure 5.2.

Oracle $\text{SIG}_1(sid, i, \mathcal{M}_1)$

01 **if** $\mathsf{Allowed}(sid, i, 1, \mathcal{M}_1) = 0$:
02 **return** $\bot$
03 $\mathsf{m} := \mathsf{message}[sid],\ S := \mathsf{signers}[sid],\ H := S \setminus \mathsf{Corrupted}$
04 $\mathsf{added} := \mathsf{AddToPending}(sid, i, \mathcal{M}_1)$ / $\mathbf{G_7}$-$\mathbf{G_{11}}$
05 **parse** $(i, S, h, \mathsf{m}, r_i) := \mathsf{state}[sid, i]$ / $\mathbf{G_0}$-$\mathbf{G_4}$
06 **parse** $(i, S, h, \mathsf{m}, \mathsf{com}_i) := \mathsf{state}[sid, i]$ / $\mathbf{G_5}$-$\mathbf{G_{11}}$
07 $\mathsf{pk}_i^{(2)} := \mathsf{T}(h, \mathsf{sk}_i)$ / $\mathbf{G_0}$-$\mathbf{G_2}$
08 $\mathsf{pk}_i^{(2)} := \mathsf{Translate}(tr[\mathsf{m}], \mathsf{pk}_i)$ / $\mathbf{G_3}$-$\mathbf{G_{11}}$
09 $r_i \xleftarrow{\$} \mathcal{D}$ / $\mathbf{G_5}$-$\mathbf{G_9}$
10 $R_i^{(1)} := \mathsf{T}(g, r_i),\ R_i^{(2)} := \mathsf{T}(h, r_i)$ / $\mathbf{G_0}$-$\mathbf{G_9}$
11 **if** $\mathsf{added} = 1$: $c := \mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
12 **if** $\mathsf{added} = 0$: $c := 0$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
13 $s_i \xleftarrow{\$} \mathcal{D}$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
14 $R_i^{(1)} := \mathsf{T}(g, s_i) - c \cdot \ell_{i,S} \cdot \mathsf{pk}_i,\ R_i^{(2)} := \mathsf{T}(h, s_i) - c \cdot \ell_{i,S} \cdot \mathsf{pk}_i^{(2)}$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
15 **if** $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] \neq \bot$: **abort** / $\mathbf{G_5}$-$\mathbf{G_{11}}$
16 $\hat{h}[S, i, R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}] := \mathsf{com}_i$ / $\mathbf{G_5}$-$\mathbf{G_{11}}$
17 $\mathsf{pm}_2[sid, i] := (R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)})$
18 $\mathsf{state}[sid, i] := (\mathcal{M}_1, (i, S, h, \mathsf{m}, r_i))$ / $\mathbf{G_0}$-$\mathbf{G_9}$
19 $\mathsf{state}[sid, i] := (\mathcal{M}_1, (i, S, h, \mathsf{m}, s_i, c))$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
20 $\mathsf{round}[sid, i] := 2$
21 $\mathsf{UpdatePending}()$ / $\mathbf{G_7}$-$\mathbf{G_{11}}$
22 **return** $\mathsf{pm}_2[sid, i]$

Figure A.15: Signing Oracle SIG_1 in the proof of Theorem 5.1. Lines with highlighted are only executed in the respective games. Algorithm Allowed is as in Figure 5.2.

Oracle $\text{SIG}_2(sid, i, \mathcal{M}_2)$

01 **if** $\text{Allowed}(sid, i, 2, \mathcal{M}_2) = 0$: **return** $\perp$
02 **parse** $(\mathcal{M}_1, (i, S, h, \mathsf{m}, r_i)) := \text{state}[sid, i]$ / $\mathbf{G_0}$-$\mathbf{G_9}$
03 **parse** $(\mathcal{M}_1, (i, S, h, \mathsf{m}, s_i, c)) := \text{state}[sid, i]$ / $\mathbf{G_{10}}$-$\mathbf{G_{11}}$
04 **parse** $(\mathsf{com}_i)_{i \in S} := \mathcal{M}_1$
05 **parse** $((R_i^{(1)}, R_i^{(2)}, \mathsf{pk}_i^{(2)}))_{i \in S} := \mathcal{M}_2$
06 **for** $j \in S$: **if** $\hat{\mathsf{H}}(S, j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) \neq \mathsf{com}_j$: **return** $\perp$
07 $R^{(1)} := \sum_{j \in S} R_j^{(1)}, \;\; R^{(2)} := \sum_{j \in S} R_j^{(2)}$
08 $\mathsf{pk}^{(2)} := \sum_{j \in S} \ell_{j,S} \mathsf{pk}_j^{(2)}$
09 $c := \bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$
10 $\text{round}[sid, i] := 3$
11 $s_i := c \cdot \ell_{i,S} \cdot \mathsf{sk}_i + r_i$ / $\mathbf{G_0}$-$\mathbf{G_9}$
12 **return** $\mathsf{pm}_3 := s_i$

Figure A.16: Signing Oracle SIG_2 in the proof of Theorem 5.1. Lines with highlighted are only executed in the respective games. Algorithm Allowed is as in Figure 5.2.

Oracle $\mathsf{H}(\mathsf{m})$

01 **if** $h[\mathsf{m}] = \perp$:
02 $\quad h[\mathsf{m}] \xleftarrow{\$} \mathcal{T}$
03 $\quad b[\mathsf{m}] \leftarrow \mathcal{B}_\gamma$ / $\mathbf{G_1}$-$\mathbf{G_{11}}$
04 $\quad$ **if** $b[\mathsf{m}] = 0$: / $\mathbf{G_2}$-$\mathbf{G_{11}}$
05 $\quad\quad (h, \mathsf{td}) \leftarrow \text{Shift}(\mathsf{par}', g)$ / $\mathbf{G_2}$-$\mathbf{G_{11}}$
06 $\quad\quad h[\mathsf{m}] := h, \; tr[\mathsf{m}] := \mathsf{td}$ / $\mathbf{G_2}$-$\mathbf{G_{11}}$
07 **return** $h[\mathsf{m}]$

Oracle $\hat{\mathsf{H}}(S, j, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)})$

08 **if** $\hat{h}[S, j, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}] = \perp$:
09 $\quad \mathsf{com} \xleftarrow{\$} \{0,1\}^{2\lambda}$
10 $\quad \hat{h}[S, j, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}] := \mathsf{com}$
11 $\quad$ **if** $\exists (sid, i, \mathcal{M}_1) \in \text{Pending}$ s.t. $\mathsf{com} \in \mathcal{M}_1$: / $\mathbf{G_7}$-$\mathbf{G_{11}}$
12 $\quad\quad$ **abort** / $\mathbf{G_7}$-$\mathbf{G_{11}}$
13 $\quad \text{UpdatePending}()$ / $\mathbf{G_7}$-$\mathbf{G_{11}}$
14 **return** $\hat{h}[S, j, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}]$

Oracle $\bar{\mathsf{H}}(\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m})$

15 **if** $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] = \perp$:
16 $\quad \bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \xleftarrow{\$} \mathcal{S}$
17 **return** $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}]$

Alg $\hat{\mathsf{H}}^{-1}(y)$

18 $P := \{x \in \{0,1\}^* \mid \hat{h}[x] = y\}$
19 **if** $|P| \neq 1$: **return** $\perp$
20 **parse** $\{x\} = P$
21 **return** x

Figure A.17: Random oracles $\mathsf{H}, \hat{\mathsf{H}}, , \bar{\mathsf{H}}$ and algorithm $\hat{\mathsf{H}}^{-1}$ in the proof of Theorem 5.1. Lines with highlighted are only executed in the respective games. Here, $\mathcal{B}_\gamma$ denotes a Bernoulli distribution with parameter $\gamma = 1/(Q_S + 1)$.

Alg AddToPending$(sid, i, \mathcal{M}_1)$

01 $S := \mathsf{signers}[sid],\ \mathsf{NotSim} := \{j \in S \mid (S, j, \mathsf{com}_j) \notin \mathsf{Sim}\}$
02 **parse** $(\mathsf{com}_j)_{j\in S} := \mathcal{M}_1$
03 **if** $\exists j \in \mathsf{NotSim}$ s.t. $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) = \bot$: **return** 0
04 **for** $j \in \mathsf{NotSim}$: $(S'_j, k_j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) := \hat{\mathsf{H}}^{-1}(\mathsf{com}_j)$
05 **if** $\exists j \in \mathsf{NotSim}$ s.t. $S'_j \neq S \vee k_j \neq j$: **return** 0
06 $\mathsf{Pending} := \mathsf{Pending} \cup \{(sid, i, \mathcal{M}_1)\}$
07 **return** 1

Alg UpdatePending$()$

08 $\mathsf{New} := \emptyset$
09 **for** $(sid, i, \mathcal{M}_1) \in \mathsf{Pending}$:
10 $\quad S := \mathsf{signers}[sid],\ \mathsf{m} := \mathsf{message}[sid]$
11 $\quad$ **parse** $(\mathsf{com}_j)_{j\in S} := \mathcal{M}_1$
12 $\quad$ **if** $\forall j \in S$: $\hat{\mathsf{H}}^{-1}(\mathsf{com}_j) \neq \bot$:
13 $\quad\quad$ **for** $j \in S$: $(S'_j, k_j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) := \hat{\mathsf{H}}^{-1}(\mathsf{com}_j)$
14 $\quad\quad$ **if** $\exists j \in S$ s.t. $S'_j \neq S \vee k_j \neq j$: **continue**
15 $\quad\quad$ $\mathsf{Pending} := \mathsf{Pending} \setminus \{(sid, i, \mathcal{M}_1)\}$
16 $\quad\quad$ $R^{(1)} := \sum_{j\in S} R_j^{(1)},\ \ R^{(2)} := \sum_{j\in S} R_j^{(2)},\ \ \mathsf{pk}^{(2)} := \sum_{j\in S} \ell_{j,S}\mathsf{pk}_j^{(2)}$
17 $\quad\quad$ **if** $(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m}) \notin \mathsf{New}$: / $\mathbf{G}_7$
18 $\quad\quad$ **if** $(S, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m}) \notin \mathsf{New}$: / $\mathbf{G}_8$-$\mathbf{G}_{11}$
19 $\quad\quad\quad$ **if** $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \neq \bot$: **abort**
20 $\quad\quad$ $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] \xleftarrow{\$} \mathcal{S}$
21 $\quad\quad$ $\bar{h}[\mathsf{pk}, \mathsf{pk}^{(2)}, R^{(1)}, R^{(2)}, \mathsf{m}] := \mathsf{GetChallenge}(sid, i, \mathcal{M}_1)$ / $\mathbf{G}_9$-$\mathbf{G}_{11}$
22 $\quad\quad$ $\mathsf{New} := \mathsf{New} \cup \{(R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})\}$ / $\mathbf{G}_7$
23 $\quad\quad$ $\mathsf{New} := \mathsf{New} \cup \{(S, R^{(1)}, R^{(2)}, \mathsf{pk}^{(2)}, \mathsf{m})\}$ / $\mathbf{G}_8$-$\mathbf{G}_{11}$

Alg Equivalent $((sid, i, \mathcal{M}_1), (sid', i', \mathcal{M}'_1))$

24 $\mathsf{m} := \mathsf{message}[sid],\ \ \mathsf{m}' := \mathsf{message}[sid'],\ \ S := \mathsf{signers}[sid],\ \ S' := \mathsf{signers}[sid']$
25 **if** $S \neq S' \vee \mathsf{m} \neq \mathsf{m}'$: **return** 0
26 **parse** $(\mathsf{com}_j)_{j\in S} := \mathcal{M}_1,\ \ (\mathsf{com}'_j)_{j\in S} := \mathcal{M}'_1$
27 $F := \{j \in S \mid \hat{\mathsf{H}}^{-1}(\mathsf{com}_j) = \bot\},\ \bar{F} := S \setminus F$
28 $F' := \{j \in S \mid \hat{\mathsf{H}}^{-1}(\mathsf{com}'_j) = \bot\},\ \bar{F}' := S \setminus F'$
29 **for** $j \in \bar{F}$: $(\tilde{S}_j, k_j, R_j^{(1)}, R_j^{(2)}, \mathsf{pk}_j^{(2)}) := \hat{\mathsf{H}}^{-1}(\mathsf{com}_j)$
30 **for** $j \in \bar{F}'$: $(\tilde{S}'_j, k_j, R_j'^{(1)}, R_j'^{(2)}, \mathsf{pk}_j'^{(2)}) := \hat{\mathsf{H}}^{-1}(\mathsf{com}'_j)$
31 $\bar{R}^{(1)} := \sum_{j\in\bar{F}} R_j^{(1)},\ \ \bar{R}^{(2)} := \sum_{j\in\bar{F}} R_j^{(2)},\ \ \bar{\mathsf{pk}}^{(2)} := \sum_{j\in\bar{F}} \ell_{j,S}\mathsf{pk}_j^{(2)}$
32 $\bar{R}'^{(1)} := \sum_{j\in\bar{F}'} R_j'^{(1)},\ \ \bar{R}'^{(2)} := \sum_{j\in\bar{F}'} R_j'^{(2)},\ \ \bar{\mathsf{pk}}'^{(2)} := \sum_{j\in\bar{F}'} \ell_{j,S}\mathsf{pk}_j'^{(2)}$
33 **if** $(\bar{R}^{(1)}, \bar{R}^{(2)}, \bar{\mathsf{pk}}^{(2)}) \neq (\bar{R}'^{(1)}, \bar{R}'^{(2)}, \bar{\mathsf{pk}}'^{(2)}) \vee (\mathsf{com}_j)_{j\in F} \neq (\mathsf{com}'_j)_{j\in F'}$: **return** 0
34 **return** 1

Alg GetChallenge$(sid, i, \mathcal{M}_1)$

35 **for** $\mathsf{rep} \in \mathsf{Reps}$: **if** $\mathsf{Equivalent}((sid, i, \mathcal{M}_1), \mathsf{rep}) = 1$: **return** $\mathsf{C}[\mathsf{rep}]$
36 $\mathsf{Reps} := \mathsf{Reps} \cup \{(sid, i, \mathcal{M}_1)\},\ \ \mathsf{C}[(sid, i, \mathcal{M}_1)] \xleftarrow{\$} \mathcal{S}$
37 **return** $\mathsf{C}[(sid, i, \mathcal{M}_1)]$

Figure A.18: Algorithms AddToPending, , UpdatePending managing list Pending, and algorithms Equivalent, GetChallenge to implement a random oracle on equivalence classes in the proof of Theorem 5.1. Lines with highlighted comments are only executed in the respective games.