Minimal Assumptions in Cryptography

Dissertation

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

von

Nils Erik Fleischhacker, Master of Science

geboren in Frankfurt am Main

Saarbrücken, 2016

Tag des Kolloquiums:	09.02.2017
Dekan:	Prof. Dr. Frank-Olaf Schreyer

Prüfungausschuss

Berichterstattende:	Prof. Dr. Dominique Schröder
	Prof. Dr. Chris Brzuska
	Prof. Dr. Abhishek Jain
Vorsitzender:	Prof. Dr. Christian Rossow
Akademischer Mitarbeiter:	Dr. Ben Stock

— For Kerstin,
always and with all my heart.

Abstract

Virtually all of modern cryptography relies on unproven assumptions. This is necessary, as the existence of cryptography would have wide ranging implications. In particular, it would hold that $\mathsf{P} \neq \mathsf{NP}$, which is not known to be true. Nevertheless, there clearly is a risk that the assumptions may be wrong. Therefore, an important field of research explores which assumptions are strictly necessary under different circumstances. This thesis contributes to this field by establishing lower bounds on the minimal assumptions in three different areas of cryptography.

We establish that assuming the existence of physically uncloneable functions (PUF), a specific kind of secure hardware, is not by itself sufficient to allow for secure two-party computation protocols without trusted setup. Specifically, we prove that unconditionally secure oblivious transfer can in general not be constructed from PUFs. Secondly, we establish a bound on the potential tightness of security proofs for Schnorr signatures. Essentially, no security proof based on virtually arbitrary non-interactive assumptions defined over an abstract group can be significantly tighter than the known, forking lemma based, proof. Thirdly, for very weak forms of program obfuscation, namely approximate indistinguishability obfuscation, we prove that they cannot exist with statistical security and computational assumptions are therefore necessary. This result holds unless the polynomial hierarchy collapses or one-way functions do not exist.

Zusammenfassung

Fast die gesamte moderne Kryptographie beruht auf unbewiesenen Annahmen. Dies ist notwendig, da die Existenz von Kryptographie weitreichende Folgen hätte. Insbesondere, müsste gelten, dass $\mathsf{P} \neq \mathsf{NP}$, was aber unbekannt ist. Dennoch besteht das Risiko, dass die Annahmen falsch sind. Deshalb ist die Untersuchung, welche Annahmen unter verschiedenen Umständen unbedingt notwendig sind, ein wichtiges Forschungsgebiet. Diese Dissertation trägt zu diesem Gebiet bei, indem untere Schranken für minimale Annahmen in drei Gebieten der Kryptographie bewiesen werden.

Wir zeigen, dass es nicht ausreicht anzunehmen, dass Physically Uncloneable Functions (PUF), eine Art von sicherer Hardware, existieren, um Protokolle für sichere Zweiparteienberechnungen ohne sogenanntes „Trusted Setup“ zu konstruieren. Genauer beweisen wir, dass bedingungslos sicherer Oblivious Transfer nicht aus PUFs konstruiert werden kann. Zweitens zeigen wir eine Schranke für die mögliche „Tightness“ von Sicherheitsbeweisen für Schnorrsignaturen. Im Wesentlichen kann kein Sicherheitsbeweis, basierend auf nahezu beliebigen nicht-interaktiven Annahmen über eine abstrakte Gruppe, signifikant „tighter“ sein als der bekannte „Forking Lemma“-Beweis. Drittens beweisen wir für sehr schwache Formen von Program Obfuscation, nämlich „Approximate Indistinguishability Obfuscation“, dass sie nicht mit statistischer Sicherheit existieren können, sondern Annahmen notwendig sind. Dieses Resultat gilt unter der Annahme, dass die polynomielle Hierarchie nicht kollabiert und One-Way-Funktionen existieren.

Background of this Dissertation

This dissertation is based on three seperate publications by the author. The papers were published at CRYPTO 2014 [DFK+14], ASIACRYPT 2014 [FJS14], and CRYPTO 2016 [BBF16b] respectively. For all three papers the author of this dissertation contributed as one of the main authors. Full versions of all three papers are publicly available on the IACR Cryptology ePrint Archive [DFK+15; FJS13; BBF16a].

[BBF16b] Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. "On Statistically Secure Obfuscation with Approximate Correctness". In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 551–578.

[FJS14] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. "On Tight Security Proofs for Schnorr Signatures". In: *Advances in Cryptology – ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, Dec. 2014, pp. 512–531.

[DFK+14] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. "Feasibility and Infeasibility of Secure Computation with Malicious PUFs". In: *Advances in Cryptology – CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 405–420.

Other Peer Reviewed Publications By the Author

[SFS+16] Jonas Schneider, Nils Fleischhacker, Dominique Schröder, and Michael Backes. "Efficient Cryptographic Password Hardening Services From Partially Oblivious Commitments". In: *ACM CCS 16: 23rd Conference on Computer and Communications Security*. Ed. by Christopher Kruegel, Andrew Myers, and Shai Halevi. Vienna, Austria: ACM Press, Oct. 2016, pp. 1192–1203.

[DFK+16] Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. "Two-Message, Oblivious Evaluation of Cryptographic Functionalities". In: *Advances in Cryptology – CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 619–648.

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. "Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys". In: *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9614. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, Heidelberg, Germany, Mar. 2016, pp. 301–330.

[FMA14] Nils Fleischhacker, Mark Manulis, and Amir Azodi. "A Modular Framework for Multi-Factor Authentication and Key Exchange". In: *SSR 2014: 1st International Conference on Research on Security Standardisation*. Ed. by Liqun Chen and Chris Mitchell. Vol. 8893. Lecture Notes in Computer Science. London, UK: Springer, Heidelberg, Germany, Dec. 2014, pp. 190–214.

[FGK+13] Nils Fleischhacker, Felix Günther, Franziskus Kiefer, Mark Manulis, and Bertram Poettering. "Pseudorandom signatures". In: *ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*. Ed. by Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng. Hangzhou, China: ACM Press, May 2013, pp. 107–118.

[FF13b] Marc Fischlin and Nils Fleischhacker. "Limitations of the Meta-reduction Technique: The Case of Schnorr Signatures". In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Athens, Greece: Springer, Heidelberg, Germany, May 2013, pp. 444–460.

[SFM+11] Immanuel Schweizer, Nils Fleischhacker, Max Mühlhäuser, and Thorsten Strufe. "SDF – Solar-Aware Distributed Flow in Wireless Sensor Networks". In: *LCN 2011: 36th IEEE Conference on Local Computer Networks*. Ed. by Tom Pfeifer, Anura Jayasumana, and Nils Aschenbruck. Bonn, Germany: IEEE, Oct. 2011, pp. 382–390.

Acknowledgements

First and foremost I thank Dominique Schröder for the opportunity of working with him, for the support during my years as a PhD student, and for helping me getting to know the amazing people that are the cryptographic community. I'm further greatly indebted to Chris Brzuska, who not only gave me the opportunity of an internship at Microsoft Research in Cambridge. The many hours of research discussions and possibly even more hours of procrastination taught me more about cryptography, research, and maybe even life than I could have imagined. I would also like to thank all of my other coauthors, in particular Dana Dachmann-Soled, Anna Lysyanskaya, Jonathan Katz, Tibor Jager, and Zvika Brakerski who worked with me on the papers that became the basis of my dissertation. I also thank Eike Kiltz, Daniel Masny, and Jiaxin Pan for pointing out a flaw in our original work on Schnorr signatures, giving me the opportunity to address this flaw in my dissertation.

The work of a researcher in theoretical computer science can be lonely and discouraging at times. Which is why I'm particularly thankful for everyone who kept me company. In particular I want to thank Mark Simkin and Johannes Krupp – my office mates before we had to move into terrariums – and Giulio Malavolta for all the discussions, the enlightening ones and especially the pointless ones that keep me sane.

I thank my family, not only because they supported me. I also sincerely thank you, for your questions of "What is it that you actually do?" which helped me stay grounded with reality. Your confusion when I try to explain keep me from forgetting that – no matter how interesting everything seems to myself – my work seems, and often quite possibly *is* utterly useless to the overwhelming majority.

And last but certainly not least, I am eternally grateful to the love of my life, my wife Kerstin. You are the most wonderful person I have ever met and at times your love and support is the only thing that keeps me going no matter what. Thank you for being you and for being there for me, I know it's probably not always be easy. I could not have done it without you.

Contents

1 Introduction

The security guarantees of any cryptographic construction stand or fall with the veracity of the assumptions made in its proof of security. Virtually all of modern cryptography crucially relies on unproven assumptions. Some cryptographic tasks can be achieved – in a limited sense – with perfect security. A prominent example of this is symmetric encryption using the One-Time Pad. However, it was already shown by Shannon [Sha49] that this requires a key that is at least as long as the message. As formally shown by Impagliazzo and Luby [IL89], even simple symmetric encryption with a short key implies the existence of one-way functions (OWF). A one-way function is a function that can be efficiently computed, but is computationally infeasible to invert, i.e., given an image under the function, it is computationally infeasible to find a preimage. One-way functions are the weakest building block of modern cryptography, but their existence would imply that $\mathsf{P} \neq \mathsf{NP}$ and would therefore solve the most prominent open question in complexity theory and maybe all of mathematics and theoretical computer science. The complexity classes P and NP are classes of decision problems, and the question whether P and NP are the same or not roughly comes down to the following: If it is possible to efficiently *verify* the solution to a problem, is it then also possible to efficiently *find* said solution? It is generally assumed that $\mathsf{P} \neq \mathsf{NP}$ but no proof is known to support this assumption. Therefore, most of modern cryptography is not even know to exist. In fact, it is very likely that $\mathsf{P} \neq \mathsf{NP}$ is not even a *sufficient* assumptions for one-way function [AGG+06; AGG+10; BB15], thus making the necessary assumptions even stronger.

While apparently necessary, basing security on unproven assumptions still seems to be a risky proposition. It is therefore important to only make those assumptions that are strictly necessary. To enable this, it is crucial to identify the *minimal assumptions*, for any application and any proof. When it is unknown what the minimal assumptions are, it is helpful to establish lower bounds on the *necessary* assumptions. To do this a number of different strategies and techniques have emerged over the last decades, each with its own benefits and problems. We outline three different strategies in the following.

Oracle Separations. In their seminal work, Impagliazzo and Rudich [IR90; IR89] proposed the oracle separation technique as a way to separate key agreement (KA) from one-way functions. In a key agreement protocol, two parties – who prior to the protocol execution share no information – interact with the goal of computing a shared secret that will remain unknown to any third party eavesdropping on the communication. The goal of Impagliazzo and Rudich was to prove that it is impossible to construct a such a protocol based solely on the assumption that one-way functions exist. However, based on the current state of knowledge, it is impossible to prove a logical statement such as $\mathsf{OWF} \implies \mathsf{KA}$, because it is not known whether key agreement exists or not. We very much hope that we live in a world where key agreement *does* exist, even if this

existing protocol may be based on something other than a one-way function. In a world where even a single secure key agreement protocol *does* exist, however, the conclusion of the statement $\mathsf{OWF} \implies \mathsf{KA}$ is already true, meaning that the statement itself is always true.

To enable a proof, Impagliazzo and Rudich, therefore, do not actually consider key agreement protocols in the real world. Instead, in a thought experiment, they construct a theoretical world, in which a single one-way function and everything that can potentially be built from this one-way function exists, but nothing else. Roughly, this world is constructed as follows: First, an oracle is introduced into this world, that can solve an NP-complete problem. This essentially means that in this world any problem in NP can now be efficiently solved by simply querying the oracle. Breaking the security of key agreement or inverting a one-way function are both problems in NP.[1] Therefore, in this world, any candidate for a one-way function can now be inverted efficiently and any candidate for a key agreement protocol can be broken. In this world neither key-agreement nor one-way functions therefore exist. In a next step, an oracle computing a uniformly chosen random function – a so called random oracle (RO) – is introduced into the world. It can be shown that with very high probability this random oracle is hard to invert. Furthermore, an oracle can always be efficiently computed by any party in this world, simply by querying it. The random oracle therefore introduces a one-way function. This means that in this theoretical world there exist only exactly those key-agreement protocols which can be built from the one-way function introduced by the random oracle. To rule out constructions of key agreement from one-way functions, Impagliazzo and Rudich next explicitly specify an efficient eavesdropper that breaks any possible key agreement protocol candidate in this world.

This seems to prove that key agreement cannot be based on the security of one-way functions alone, since otherwise a key agreement protocol would exist in this world. However, there is a caveat to this result, because it implicitly restricts the constructions of a key agreement protocol to be "black-box". A construction is said to be *black-box* in the underlying primitive, if it only depends on the input/output behavior of the primitive. A construction that would be *non*-black-box could for example depend on the internal description – the source code – of the primitive. In the theoretical world constructed by Impagliazzo and Rudich, such a non-black box construction is impossible, because the primitive does not even have a short description. It is therefore left open, whether non-black-box techniques may be able to circumvent the result. The proof of Impagliazzo and Rudich was later refined by Barak and Mahmoody [BM09; BM08] giving a tighter analysis. The same oracle separation technique and refinements [HR04] of it were also used to establish separations between collision resistant hash functions and one-way functions [Sim98], oblivious transfer and public key encryption [GKM+00] and many more [GMR01; CHL02; Fis02; DOP05; BCF+09; FLR+10; Hof11; Rud92; KSY11; FS12]. This technique is also used in this thesis in Chapter 2 to separate oblivious transfer and secure two-party computation from malicious physically uncloneable functions, i.e., from the assumption that a special kind of secure hardware exists.

Meta-Reductions. An alternative separation technique, called meta-reductions, was first introduced by Boneh and Venkatesan [BV98], although the term meta-reduction was coined in later works [PV05; Bro05; Bro16]. Meta-reductions are most commonly used to separate more

[1] More precisely, the corresponding decision problems are in NP and a decider for the decision problems can be used to solve the computational problem.

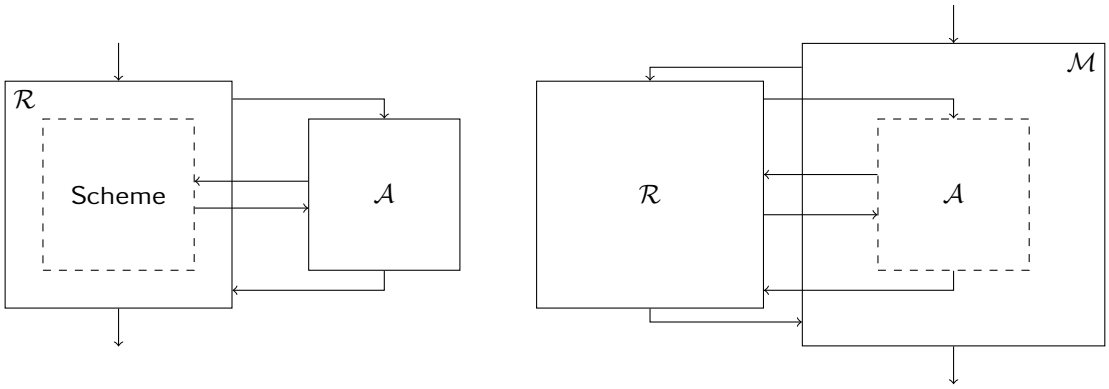


Figure 1.1: Depicted on the left hand side is the basic idea of a security reduction: The reduction gets as input some problem instance and solves it by interacting with an adversary while simulating the interaction the adversary would have with the cryptographic scheme. Depicted on the right hand side is the basic idea of a meta-reduction: The meta-reduction gets as input some problem instance and solves it by interacting with a reduction while simulating the interaction the reduction would have with the adversary.

specific assumptions and schemes. E.g., Boneh and Venkatesan [BV98] use it to separate the RSA assumption from the assumption that factoring is hard. However, it can also be used in more general cases, such as ruling out blind signature schemes with less than four communication-rounds based on any non-interactive assumption [FS10]. The basic idea of a meta-reduction is to construct a "reduction against the reduction".

A proof by reduction is the most prevalent technique for proving the security of a cryptographic construction. Conceptually, this works as follows: To prove that a construction is secure, one first needs to assume that some underlying problem, that we will call Π, is hard. For example, we might assume that a function is a one-way function, i.e., we assume that the function is hard to invert. We then assume that a successful attacker $\mathcal{A}$ against our construction exists and bring the two assumptions to a contradiction. This is done by constructing a *reduction* $\mathcal{R}$ that uses its interaction with $\mathcal{A}$ to solve the problem Π. This is depicted on the left side of Figure 1.1. The reduction $\mathcal{R}$ get as input an instance of the problem Π, e.g., for inverting a one-way function this would simply be an image under the function. It then interacts with $\mathcal{A}$, simulating the construction that $\mathcal{A}$ is supposedly able to attack. In this interaction, $\mathcal{R}$ would in some way embed the problem instance in such a way, that the output of the attacker would allow the reduction to correctly solve the problem instance.

A meta-reduction uses essentially the same concept, but against the reduction instead of the attacker. I.e., we assume that there exists a reduction $\mathcal{R}$ supposedly able to solve a hard problem Π when given access to any attacker $\mathcal{A}$ able to break the construction and once again bring the assumptions that Π is hard and that $\mathcal{R}$ exists to a contradiction. This is done by constructing a *meta-reduction* $\mathcal{M}$ that uses $\mathcal{R}$ to *directly* solve the problem Π. This is depicted on the right side of Figure 1.1. The meta-reduction $\mathcal{M}$ get as input an instance of the problem Π. It then interacts with $\mathcal{R}$ and simulates an attacker $\mathcal{A}$, thus turning $\mathcal{R}$ into a successful algorithm $\mathcal{M}^{\mathcal{R}}$ for solving the problem Π. The key part of this technique is of course the simulation of $\mathcal{A}$.

Clearly, a successful meta-reduction requires two things. First, a successful attacker $\mathcal{A}$ against the construction must exists in the first place. Otherwise $\mathcal{R}$ does not need to be successful at all. Second, $\mathcal{M}$'s simulation of $\mathcal{A}$ must be "convincing". This seems to lead to a contradiction. If a successful efficient attacker is known to exist, then the construction is obviously already insecure. Therefore, there would be no point in proving that its security cannot be based on some assumption. To circumvent this seeming contradiction, we must assume that $\mathcal{R}$ is fully *black-box* in the attacker. Essentially, this means that $\mathcal{R}$ must be successful when given access to any algorithm having the correct input-output behavior of a successful attacker. This in particular includes *inefficient* attackers. For most cryptographic constructions, inefficient attackers trivially exist, for example attackers that use exhaustive search over all possible keys to break an encryption scheme. Note, that this does not imply that the construction is insecure, because computational security definitions only consider *efficient* attackers.

The meta-reduction $\mathcal{M}$ must of course still be able to efficiently *simulate* $\mathcal{A}$, which may again seem like a contradiction. If an inefficient attacker can be simulated efficiently, then seemingly an efficient attacker also exists. However, this is not true. The meta-reduction commonly manages to efficiently simulate an inefficient attacker by leveraging additional capabilities that an attacker would not possess. This includes for example the ability of $\mathcal{M}$ to reset or rewind the reduction $\mathcal{R}$ to an earlier state. Another approach is to have $\mathcal{M}$ actually reduce to a different problem Π' than the problem Π, the reduction reduces too. This is helpful if Π' gives more power to an algorithm trying to solve it than Π. An example for this would be if Π is the discrete logarithm problem and Π' is the *one-more* discrete logarithm problem. If $\mathcal{M}$ manages to simulate $\mathcal{A}$ in such a way that the interaction of $\mathcal{R}$ with the simulated attacker is indistinguishable from the interaction with the actual (inefficient) $\mathcal{A}$, then $\mathcal{M}^{\mathcal{R}}$ can leverage the success probability of $\mathcal{R}^{\mathcal{A}}$ to directly solve the problem Π or Π'.

Meta-reductions are a very powerful separation technique. In particular, in contrast to oracle separations described above, meta-reduction usually do *not* require the construction or the security reduction to be black-box in the underlying primitive. As such they have been successfully applied to achieve a great number of results such as [Cor02b; PV05; FS10; Pas11; GW11; DHT12; Seu12; BL13a; FF13b; ZZC+14; CHZ14; ZCC+15; BJL+16; BFW16]. It is also noteworthy that separations via meta-reduction are not necessarily unconditional, but can in fact leverage additional assumptions. In the high level description above, this is exactly the case if $\mathcal{M}$ solves a different problem Π' from the problem Π that the reduction solves. The interpretation of this case would be that a security reduction to problem Π cannot not exist *if* Π' *is a hard problem*. Examples of this include the works of Paillier and Vergnaud [PV05] and of Fischlin and Fleischhacker [FF13b] who both rule out certain classes of security reductions for Schnorr signatures to the discrete logarithm problem, under the assumption that the weaker *one-more* discrete logarithm problem is hard.

The meta-reduction technique is also used in this thesis in Chapter 3 to establish several lower bounds on the minimal assumptions required to prove security for Schnorr's signature scheme. These bounds are in some instances orthogonal to and in other instances greatly extend results about Schnorr signatures mentioned above.

Contradictions to Commonly Held Beliefs. A different strategy is to show that the existence of some cryptographic primitive or proof thereof would lead to a contradiction with other assumptions that – while not strictly proven – are commonly accepted as true by researchers

in the relevant fields. Most commonly these are assumptions from complexity theory, such as $\mathsf{P} \neq \mathsf{NP}$ or the generalization thereof, i.e., the assumption that the so-called polynomial hierarchy does not collapse. Insofar, this type of result does not necessarily show that something cannot possibly exist, but instead that the existence would have far reaching consequences. Essentially, if the primitive or proof exists, then the world we live in differs fundamentally from the world we commonly think we live in.

This strategy can also be particularly helpful to show that a certain primitive cannot exists with *unconditional* security. While such a result may seem qualitatively different from the separation results discussed before, it can nevertheless often be seen as a lower bound on necessary assumptions. I.e., it shows a lower bound in the sense that at least *some* assumption is necessary to construct the primitive.

Early examples include the work of Brassard [Bra79] and the later clarification thereof [GG98] showing that the security of certain special types of public key encryption cannot be based on NP-hardness unless $\mathsf{NP} = \mathsf{coNP}$ which would imply the collapse of the polynomial hierarchy. Further negative results of this kind found that is is unlikely to reduce the average-case complexity of an NP problem to the worst-case complexity of an NP problems [BT03; BT06] or to base the security of certain types of one-way functions [AGG+06; AGG+10; BB15] or private information retrieval [LV16] on NP-hardness. Examples of impossibility results for statistically secure cryptography include proofs that neither statistically sound witness encryption [GGS+13] nor statistically secure indistinguishability obfuscation [GR07] exists unless the polynomial hierarchy collapses.

With the introduction of many new extremely powerful cryptographic primitives, a new flavor of this kind of result has also recently emerged. These so called one-out-of-two results do not show that the existence of one primitive has unlikely implications about the landscape of complexity theory, but instead shows that two different primitives cannot coexist. For example, the existence of indistinguishability obfuscation implies attacks against certain types of Universal Computational Extractors [BFM14], obfuscation of multi-bit point functions with auxiliary input [BM14], and extractable one-way functions [BCP+14].

In Chapter 4 a sort of hybrid of these two types of proofs is employed to show that an extremely weak form of obfuscation (approximate correlation obfuscation) cannot exist with statistical security, unless either one-way functions do not exist or the polynomial hierarchy collapses.

1.1 Contribution

This thesis establishes lower bounds for minimal assumptions in in three areas of cryptography. These lower bounds differ in their nature, quality and in the way they are achieved.

In Chapter 2 we use an oracle separation technique to give a lower bound for the necessary assumptions of oblivious transfer and general secure two-party computation. In particular, we show that oblivious transfer, and by implication secure two-party computation, can in general not be based on the security of a type of secure hardware known as physically uncloneable functions (PUFs) in a black-box way. For general PUFs, this result holds unconditionally. However, we also show how this result *can* be sidestepped by making additional assumptions on the power of the attacker. Whether these assumptions might be reasonable, largely depends on the precise instantiation of PUFs.

In Chapter 3 we use a meta-reduction to establish lower bounds for the minimal assumption necessary for proving the security of Schnorr signatures. Our main result shows that the security of Schnorr signatures cannot be *tightly* reduced to almost any natural non-interactive assumption. This means that a *much harder* underlying problem needs to be assumed to achieve any concrete provable hardness of forging Schnorr signatures. Thus it gives a lower bound on the necessary hardness of the underlying problem.

In Chapter 4 we show that it is very unlikely that even extremely weak forms of program obfuscation can be instantiated *with statistical security*. In particular, we show that the existence of statistically secure approximate indistinguishability obfuscation (saiO) would imply the collapse of the polynomial hierarchy. It thus contradicts the commonly accepted complexity theoretic assumption that the polynomial hierarchy does not collapse. The result therefore shows that, unless the polynomial hierarchy collapses, it is necessary to make computational assumptions when instantiating approximate indistinguishability obfuscation.

1.1.1 Secure Computation Based on Secure Hardware

Secure two party computation in general refers to a protocol, in which two parties interact to jointly evaluate some functionality. Both parties may have input to the functionality and both parties may receive output. Intuitively, the security guarantee of such a protocol is that neither party learns more about the other parties input than can be trivially inferred from the received output. Oblivious transfer is one of the major building blocks of secure two-party computation for arbitrary functionalities. In fact, assuming oblivious transfer is sufficient for constructing secure two-party computation for arbitrary functionalities. In an oblivious transfer protocol, party A holds two secrets s_0, s_1 and party B holds a choice bit b. After the execution of the protocol, B should learn s_b but learn nothing about s_{1-b}, whereas A should be unable to tell which secret B learned.

Secure two-party computation with strong security guarantees was shown by Canetti and Fischlin [CF01] to be impossible for arbitrary functionalities in the so called "plain model" where no infrastructure besides communication channels exists. In fact Canetti et al. [CKL06] later showed that the impossibility holds for essentially all non-trivial functionalities. An interesting question was therefore whether this impossibility could be sidestepped by assuming some kind of setup. It turns out that this is indeed possible, at least using *trusted* setup. Examples of sufficient trusted setup include a common reference string [CF01; CLO+02] or a trusted public keys registration service [BCN+04; CDP+07].

The idea of using *secure hardware* to replace trusted parties as a setup assumption was first explored by Katz [Kat07] who showed that tamper-proof hardware tokens are sufficient to circumvent the impossibility results. A different kind of secure hardware that has been proposed are *physically uncloneable functions* (PUF) [Pap01; PRT+02; MV10; AMS+11; KKR+12]. A PUF is supposed to be a physical object generated via a process that is intended to create "unique" objects that can be probed and whose responses behave like a random function. I.e., a PUF is supposed to look *random*, and be *impossible to copy* even by the entity who created the PUF. PUFs seem to be extremely powerful and in particular Brzuska et al. [BFS+11] showed that PUFs allowed for universally composable constructions of bit commitment, key agreement, and oblivious transfer (and hence secure computation of arbitrary functionalities) with *unconditional* security. They, thus, also seemed to be a good candidate for a physical setup assumption that

does not require trusted parties. However it was observed by Ostrovsky et al. [OSV+13], that the result of Brzuska et al. implicitly assumes that all PUFs – including those created by the attacker – are honestly generated. This means that the model implicitly reintroduce an assumption of trust, namely the trust that PUFs are created honestly – possibly by some designated trusted party. The question whether PUFs without an additional assumption of trust still allow *unconditionally* secure computation of arbitrary functionalities was explicitly left open by Ostrovsky et al. and remained unanswered since.

In Chapter 2 we answer this question negatively. Our proof specifies an oracle separation showing that oblivious transfer cannot be based on maliciously generated PUFs in a black-box way. Our impossibility result crucially relies on malicious PUFs being able to keep state. In Section 2.4 we show that this is indeed inherent, since it is possible to achieve unconditionally secure computation of arbitrary functionalities if malicious PUFs can be guaranteed to be stateless.

1.1.2 Tight Security of Schnorr Signatures

Schnorr's signature scheme is one of the most fundamental public-key cryptosystems and was proven secure by Pointcheval and Stern based on the generally accepted discrete logarithm assumption [PS96] in the Random Oracle Model (ROM) [BR93]. However, their reduction is not tight. This means that the reduction is significantly less successful in solving the discrete logarithm problem than the attacker is in breaking the signature scheme. In particular, if the attacker successfully forges a signature with probability ϵ after making q queries to the random oracle, then the reduction only solves the discrete logarithm problem with probability ϵ/q.

In an asymptotic sense this is still a valid security reduction, since for any non-negligible function ϵ and any polynomial number of queries q, the function ϵ/q is also non-negligible. However in any concrete setting, such a reduction is almost useless for determining secure parameters for the underlying discrete logarithm problem because the underlying problem must be at least q times as hard as the signature scheme. Since q depends on the run-time of the attacker, but the hardness of the discrete logarithm problem is constant, a large constant upper bound (say 2^{80} or 2^{100}) for q must be chosen. While this might lead to a secure choice of parameters, the resulting keysize would be astronomical. We therefore investigate whether a tight security reduction for Schnorr signatures is possible.

Previous work [PV05; GBL08; Seu12] in this area showed that with respect to a limited (but natural) class of reductions (so-called *algebraic reductions*) the proof of Pointcheval and Stern is essentially optimal under the assumption that the stronger one-more discrete logarithm problem [BNP+03] is hard. While these impossibility results are already quite general, they are explicitly limited to the case of reducing to the discrete logarithm problem. I.e., they do not make any statement whether tight security reductions to other, *stronger* assumptions, such as the computational or decisional Diffie-Hellman assumptions, might exist. In Chapter 3 we answer this question in the negative. Our impossibility result rules out the existence of tight reductions for virtually all natural non-interactive computational problems defined over abstract algebraic groups. Moreover – in contrast to previous works – our results hold *unconditionally*. On a technical level, our result is incomparable with previous work, because we need to limit the class of reductions slightly more than previous work. I.e., our result holds for *generic* reductions instead of algebraic ones.

Algebraic and generic algorithms both are constrained in how they can use group elements. In essence, in both cases it is enforced that the algorithm can only act upon group elements by using the defined group operation. Technically, this is done in two different ways, which leads to the main difference. While both kinds of algorithm can only use group operations to act upon group elements, an algebraic algorithm can make *decisions based on the representation* of group elements, whereas a generic algorithm cannot. Therefore, an algebraic reduction is more powerful and, thus, considering only generic reductions is more restrictive.

However, the main motivation of considering only algebraic reductions is that non-algebraic ones are essentially unknown. I.e., all known security reductions for discrete logarithm based cryptographic constructions *are* algebraic. It turns out, that this motivation applies equally to generic reductions. To the best of our knowledge all known examples of algebraic reductions are also generic.

The proof technique uses meta-reductions in combination with a novel simulation strategy. In Section 3.5 we also show that the new simulation strategy can be applied in a slightly different context, ruling out any (even non-tight) generic security proof in a weaker model known as the non-programmable random oracle model (NPROM). The NPROM is strictly weaker than the ROM. While in the ROM the reduction fully controls the random oracle and can in particular reprogram parts of the oracle on the fly, the random oracle is external to the reduction in the NPROM. This means that the reduction can *observe* the queries the attacker makes to the random oracle, but it cannot *influence* the answers. A security proof in a weaker model would be a stronger argument for security of the scheme.

1.1.3 Statistically Secure Obfuscation

Program obfuscation in general is the concept of taking a program and transforming it in such a way that the resulting program is completely unintelligible but nevertheless still computes the same function. A very natural formalization of this concept is that the obfuscated code should not reveal anything that cannot also be learned from access to an oracle containing the program. Barak et al. [BGI+01; BGI+12], however, showed that this strong kind of obfuscator does not exist in general. A much weaker form obfuscation called Indistinguishability Obfuscation (iO) was proposed. An indistinguishability obfuscator only guarantees, that if there are two programs that compute exactly the same function (and are of the same size) then obfuscations of the two are indistinguishable. Intuitively this security guarantee appears very weak and one might even suspect that such obfuscators would be useless. However the opposite turned out to be true. Since the proposal of the first candidate construction by Garg et al. [GGH+13], iO has firmly established itself as a very powerful tool. It is also one of the few known non-black-box techniques for the construction of cryptographic objects. As such, iO, e.g., allows to transform symmetric encryption into public-key encryption [SW14]. Therefore, the existence of a *statistically secure indistinguishability obfuscator* (siO) would sidestep the black-box impossibility discussed above and resolve the question of constructing public key cryptography from one-way functions. This even remains true for weaker forms of obfuscation that only preserve *approximate* correctness. I.e., the symmetric to public key transformation of Sahai and Waters [SW14] does not require that the obfuscated program computes exactly the same function. It is instead sufficient that there is a non-negligible correlation between the functionality of the input program and that of the output program. We are therefore interested in answering the question whether

obfuscation with statistical security may exist.

Previous work in this area by Goldwasser and Rothblum [GR07; GR14] could only rule out statistically secure obfuscation with *perfect* correctness under the assumption that the polynomial hierarchy does not collapse. While this result already put a damper on hopes to achieve statistically secure obfuscation, it has no known implications for the existence of statistically secure obfuscation with *approximate* correctness (saiO).

In Chapter 4 we prove that it is very unlikely that saiO exists. In particular, we show that saiO does not exist if one-way functions exist and the polynomial hierarchy does not collapse. Our impossibility results extend even beyond the case of saiO. In fact, the result applies even when the *security* of the obfuscator is approximate. Namely, when we are only guaranteed that the obfuscation of functionally equivalent circuits results in distributions that have mild statistical distance (as opposed to negligible). While the impossibility result extends to a large range of parameters, it turns out that we cannot rule out *all* useful parameters, thus still leaving a gap for a potential non-black-box construction of public-key encryption from one-way functions.

1.2 Notation

We introduce some general notation. By $n \in \mathbb{N}$, we denote the security parameter that we give to all algorithms implicitly in unary representation 1^n. By $\{0,1\}^\ell$ we denote the set of all bit-strings of length ℓ. For a finite set S, we denote the action of sampling x uniformly at random from S by $x \leftarrow\!\!\$\, S$, and denote the cardinality of S by $|S|$. We call an algorithm efficient or PPT (probabilistic polynomial time) if it runs in time polynomial in the security parameter. Unless explicitly stated to be deterministic, all algorithms are assumed to be randomized. If $\mathcal{A}$ is randomized then by $y \leftarrow \mathcal{A}(x;r)$ we denote that $\mathcal{A}$ is run on input x and with random coins r and produces output y. If no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly sampled uniform random coins, and write this as $y \leftarrow\!\!\$\, \mathcal{A}(x;\mathcal{U})$, where $\mathcal{U}$ refers to the uniform distribution over random coins, or in shorthand $y \leftarrow\!\!\$\, \mathcal{A}(x)$. We denote with $\emptyset$ the empty string, the empty set, as well as the empty list, the meaning will always be clear from the context. We write $[n]$ to denote the set of integers from 1 to n, i.e., $[n] := \{1,\ldots,n\}$. For a circuit C we denote by $|C|$ the size of the circuit, i.e., the number of gates. We say a function $\mathsf{negl}(n)$ is negligible if for any positive polynomial $\mathsf{poly}(n)$, there exists an $N \in \mathbb{N}$, such that for all $n > N$, $\mathsf{negl}(n) \leq \frac{1}{\mathsf{poly}(n)}$.

We will use the following definition of statistical distance.

Definition 1 (Statistical Distance)**.** *For two probability distributions X, Y we define the statistical distance* $\mathsf{SD}(X,Y)$ *as*

$$\mathsf{SD}(X,Y) = \max_{\mathcal{A}} \Big(\Pr_{x \leftarrow\!\!\$\, X}[\mathcal{A}(x) = 1] - \Pr_{y \leftarrow\!\!\$\, Y}[\mathcal{A}(y) = 1] \Big)$$

where $\mathcal{A}$ ranges over all probabilistic algorithms including inefficient ones.

2 On Secure Computation with Malicious PUFs

2.1 Introduction

A *physically uncloneable function* (PUF) [Pap01; PRT+02; MV10; AMS+11; KKR+12] is a physical object generated via a process that is intended to create "unique" objects with "random" (or at least random-looking) behavior. PUFs can be probed and their response measured, and a PUF thus defines a function. (We ignore here the possibility of slight variability in the response, which can be corrected using standard techniques.) At an abstract level, this function has two important properties: it is *random*, and it *cannot be copied* even by the entity who created the PUF.

Since their introduction, several cryptographic applications of PUFs have been suggested, in particular in the area of secure computation. PUFs are especially interesting in this setting because they can potentially be used (1) to obtain *universally composable* (UC) protocols [Can01] without additional setup, thus bypassing known impossibility results that hold for universal composition in the "plain" model [CF01; CKL06], and (2) to construct protocols with *unconditional* security, i.e., without relying on any cryptographic assumptions.

Initial results in this setting [Rüh10; RKB10] showed constructions of oblivious transfer with stand-alone security based on PUFs. Brzuska et al. [BFS+11] later formalized PUFs within the UC framework, and showed UC constructions of bit commitment, key agreement, and oblivious transfer (and hence secure computation of arbitrary functionalities) with *unconditional* security. The basic feasibility questions related to PUFs thus seemed to have been resolved.

Ostrovsky et al. [OSV+13], however, observe that the previous results implicitly assume that all PUFs, including those created by the attacker, are honestly generated. They point out, correctly, that this may not be a reasonable assumption: nothing forces the attacker to use the recommended process for manufacturing PUFs and it is not clear, in general, how to "test" whether a PUF sent by some party was generated correctly or not. (Assuming a trusted entity who creates the PUFs is not a panacea, as one of the goals of using PUFs is to avoid reliance on trusted parties.) Addressing this limitation, Ostrovsky et al. define a model in which an attacker can create *malicious* PUFs having arbitrary, adversary-specified behavior. The previous protocols can be easily attacked in this new adversarial setting, but Ostrovsky et al. show that it is possible to construct universally composable protocols for secure computation in the malicious-PUF model under additional, number-theoretic assumptions. They explicitly leave open the question of whether unconditional security is possible in the malicious-PUF model. Recently, Damgård and Scafuro [DS13] have made partial progress on this question by presenting a commitment scheme with unconditional security in the malicious-PUF model.

Stateful vs. stateless (malicious) PUFs. Honestly generated PUFs are stateless; that is, the output of an honestly generated PUF is independent of its computation history. Ostrovsky et al. note that maliciously generated PUFs might be stateful or stateless. Allowing the adversary to create stateful PUFs is obviously more general. (The positive results mentioned earlier remain secure even against an attacker who can create malicious, stateful PUFs.) Nevertheless, the assumption that the adversary is limited to producing stateless PUFs is meaningful; indeed, depending on the physical technology used to implement the PUFs, incorporating dynamic state in the PUF may simply be infeasible.

2.1.1 Our Results

Spurred by the work of Ostrovsky et al. and Damgård and Scafuro, we reconsider the possibility of unconditionally secure computation based on malicious PUFs and resolve the main open questions in this setting. Specifically, we show:

1. Unconditionally secure oblivious transfer (and thus unconditionally secure computation of general functions) is impossible when the attacker can create malicious *stateful* PUFs. Our result holds even with regard to stand-alone security, and even for indistinguishability-based (as opposed to simulation-based) security notions.

2. If the attacker is limited to creating malicious, but *stateless*, PUFs, then universally composable oblivious transfer (OT) and two-party computation of general functionalities are possible. Our oblivious-transfer protocol is efficient and requires each party to create only a single PUF for polynomially many OT executions. The protocol is also conceptually simple, which we view as positive in light of the heavy machinery used in [OSV+13].

2.1.2 Comparison to [DFK+14]

We extend and improve the proceedings version [DFK+14] in many respects. First, we identify several pitfalls in formalizing PUFs and provide a corrected definition that addresses these issues. For example, [OSV+13] does not require that the evaluation algorithm of the (malicious) PUF runs in polynomial-time. The basic idea here is that the evaluation algorithm performs some physical operations that cannot be computed efficiently. However, simulation in this case may become difficult because the PUF may simply solve a hard problem, such as factoring, internally before outputting the response. Another difference to [DFK+14] is that we provide a simplified protocol for our positive result. That is, the OT-protocol suggested in [DFK+14] requires two PUFs, while the protocol in this version only requires a single PUF. Finally, this version contains all proofs, while parts of the proofs in [DFK+14] were omitted.

2.1.3 Other Related Work

Hardware tokens have also been proposed as a physical assumption on which to base secure computation [Kat07]. PUFs are incomparable to hardware tokens since they are more powerful in one respect and less powerful in another. PUFs have the property that a party cannot query an honestly generated PUF when it is out of that party's possession, whereas in the token model parties place known functionality in the token and can simulate the behavior of the token at any point. On the other hand, tokens can implement arbitrary code, whereas honestly generated

PUFs just provide a random function. In any case, known results (such as the fact that UC oblivious transfer is impossible with stateless tokens [GIM+10]) do not directly translate from one model to the other.

Impossibility results for (malicious) PUFs are also not implied by impossibility results in the random-oracle model (e.g., [BM09]). A random oracle can be queried by any party at any time, whereas (as noted above) an honestly generated PUF can only be queried by the party who currently holds it. Indeed, we show that oblivious transfer *is* possible when malicious PUFs are assumed to be stateless; in contrast, oblivious transfer is impossible in the random-oracle model [IR89].

Ostrovsky et al. [OSV+13] consider a second malicious model where the attacker can *query* honestly generated PUFs in a non-prescribed manner. They show that secure computation is impossible if both this and maliciously generated PUFs are allowed. We do not consider the possibility of malicious queries in this work.

In other work, Rührmair and van Dijk [RD13] show impossibility results in a malicious-PUF model that differs significantly from the ones considered in [OSV+13; DS13] and here. Their model appears to be strictly weaker than the model of Ostrovsky et al., i.e., it seems to guarantee security against strictly more powerful adversaries. However, it is not entirely clear to us to which extent the additional power given to the attackers corresponds to attacks that could feasibly be carried out in the real world. In [DR12] van Dijk and Rührmair informally discussed the idea of using the technique of Impagliazzo and Rudich [IR89] in the context of PUFs. Although the authors claim to provide a formal impossibility result, they do not give a formal definition of what a "bad PUF" is. Moreover, applying the technique of of Impagliazzo and Rudich [IR89] in the context of PUFs does not straightforwardly seem to be possible, as we show in this work.

2.2 Impossibility Result for Malicious, Stateful PUFs

A physically uncloneable function (PUF) is a physical device with "random" behavior introduced through uncontrollable manufacturing variations during their fabrication. When a PUF is queried with a stimulus (i.e., a challenge), it produces a physical output (the response). In practice, the output of a PUF can be noisy; i.e., querying the PUF twice with the same challenge may yield distinct, but close, responses. Moreover, the response need not be uniform; it may instead only have high min-entropy. For the purpose of the impossibility result, however we will skip a formal definition of PUFs including all these subtleties. Instead, for the purpose of the impossibility result, we will work with theoretical ideal PUFs.

Namely, an honestly generated ideal PUF for the purpose of this proof is a piece of hardware that implements a truly random function and allows only black-box access to this function. A maliciously generated PUF in this ideal setting, can contain any efficiently computable program specified by the generating party.

Similarly, we consider an indistinguishability based definitions of security for OT and not a UC definition of OT. Since we are going to show an impossibility result, using ideal PUFs as the building block and ruling out weaker security notions only makes our result stronger (since a noisy PUF can be simulated given an ideal PUF by querying the ideal PUF and adding noise). We prove that any PUF-based oblivious-transfer (OT) protocol is insecure when the attacker has the ability to generate malicious, *stateful* PUFs. Formally:

Theorem 2. *Let Π be a PUF-based OT-protocol where the sender $\mathcal{S}$ and receiver $\mathcal{R}$ each make at most $m = \mathsf{poly}(n)$ PUF queries. Then at least one of the following holds:*

1. *There is an unbounded adversary $\mathcal{S}^*$ that uses malicious, stateful PUFs, makes only $\mathsf{poly}(n)$ queries to honestly generated PUFs, and computes the choice bit of $\mathcal{R}$ (when $\mathcal{R}$'s input is uniform) with probability $1/2 + 1/\mathsf{poly}(n)$.*

2. *There is an unbounded adversary $\mathcal{R}^*$ that uses malicious, stateful PUFs, makes only $\mathsf{poly}(n)$ queries to honestly generated PUFs, and correctly guess both secrets of $\mathcal{S}$ (when $\mathcal{S}$'s inputs are uniform) with probability at least $2/3$.*

2.2.1 Overview

The starting point for our impossibility result is the impossibility of constructing oblivious transfer in the random-oracle model. The fact that OT is impossible in the random-oracle model follows from the fact that key agreement is impossible in the random-oracle model [IR89; BM09; BM13], and the observation that OT implies key agreement. However, a direct proof ruling out OT in the random-oracle model is also possible, and we sketch such a proof here.

Consider an OT protocol in the random-oracle model between a sender $\mathcal{S}$ and receiver $\mathcal{R}$, where $\mathcal{S}$'s two input bits are uniform and $\mathcal{R}$'s selection bit is uniform. We show that either $\mathcal{S}$ or $\mathcal{R}$ can attack the protocol. Consider the case where both parties run the protocol honestly and then at the end of the protocol they each run a variant of the Eve algorithm from [BM09; BM13] to obtain a set Q of queries/answers to/from the random oracle. This set Q contains all "intersection queries" between $\mathcal{S}$ and $\mathcal{R}$, which are queries made by both parties to the random oracle. However, note that the setting here is different from the key-agreement setting in which a third party (the eavesdropper) runs the Eve algorithm. In fact, in our setting, finding intersection queries is trivial for $\mathcal{S}$ and $\mathcal{R}$: all intersection queries are, by definition, already contained in the view of either of the parties. Thus, the point of running the Eve algorithm is for both parties to reconstruct the *same* set of queries Q that contains all intersection queries. As in [BM09; BM13], conditioned on the transcript of the protocol and this set Q, the views of $\mathcal{S}$ and $\mathcal{R}$ are independent. The property of the Eve algorithm we use is that with high probability over random coins of the protocol and the choice of the random oracle, the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Q is statistically close to the distribution over $\mathcal{R}$'s view conditioned on only the transcript and Q.

To use the above to obtain an attack, we first consider the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Q. We argue that with probability roughly $1/2$ over this distribution, $\mathcal{R}$'s view must be consistent with selection bit 0, and with probability $1/2$ it must be consistent with selection bit 1. (If not, then $\mathcal{S}$ can compromise $\mathcal{R}$'s security by guessing that $\mathcal{R}$'s selection bit is the one which is more likely.) Next, we consider the distribution over $\mathcal{R}$'s view conditioned on only the transcript and Q. Note that $\mathcal{R}$ can sample from this distribution, since $\mathcal{R}$ knows the transcript and can compute the same set Q. Since this distribution is statistically close to the distribution over $\mathcal{R}$'s view conditioned on $\mathcal{S}$'s view and Eve queries, we have that $\mathcal{R}$ can with high probability sample a view consistent with selection bit 0 and $\mathcal{S}$'s view *and* a view consistent with selection bit 1 and $\mathcal{S}$'s view. But correctness of the protocol then implies that $\mathcal{R}$ can with high probability discover both of $\mathcal{S}$'s inputs.

From random oracles to PUFs. The problem with extending the above to the PUF model is that, unlike a random oracle, a PUF can only be queried by the party who currently holds it. This mean that the above attack, as described, will not work. In fact, this property is what allows us to *construct* an OT protocol in the case where malicious PUFs are assumed to be stateless! To overcome this difficulty, we will need to use the fact that malicious parties can create *stateful* PUFs.

To illustrate the main ideas, consider a protocol in which four PUFs are used. $\mathsf{PUF}_{\mathcal{S}}$ and $\mathsf{PUF}'_{\mathcal{S}}$ are created by $\mathcal{S}$, with $\mathsf{PUF}_{\mathcal{S}}$ held by $\mathcal{S}$ at the end of the protocol and $\mathsf{PUF}'_{\mathcal{S}}$ held by $\mathcal{R}$ at the end of the protocol. Similarly, $\mathsf{PUF}_{\mathcal{R}}, \mathsf{PUF}'_{\mathcal{R}}$ are created by $\mathcal{R}$, with $\mathsf{PUF}_{\mathcal{R}}$ held by $\mathcal{R}$ at the end of the protocol and $\mathsf{PUF}'_{\mathcal{R}}$ held by $\mathcal{S}$ at the end of the protocol. We now want to provide a way for both parties to be able to obtain a set Q of queries/answers for all the PUFs that contains the following "intersection queries":

1. Any query that *both* parties made to $\mathsf{PUF}'_{\mathcal{S}}$ or $\mathsf{PUF}'_{\mathcal{R}}$ (as in [BM09; BM13]).

2. All queries that $\mathcal{R}$ made to $\mathsf{PUF}_{\mathcal{S}}$.

3. All queries that $\mathcal{S}$ made to $\mathsf{PUF}_{\mathcal{R}}$.

The first of these can be achieved by having $\mathcal{S}$ (resp., $\mathcal{R}$) construct $\mathsf{PUF}'_{\mathcal{S}}$ (resp., $\mathsf{PUF}'_{\mathcal{R}}$) with known code, such that $\mathcal{S}$ (resp., $\mathcal{R}$) can effectively query $\mathsf{PUF}'_{\mathcal{S}}$ (resp., $\mathsf{PUF}'_{\mathcal{R}}$) at any time. Formally, we have each party embed a randomly chosen t-wise independent function in the PUF they create, where t is large enough so that the behavior of the PUF is indistinguishable from a random function as far as execution of the protocol (and the attack) is concerned. At the end of the protocol, both parties can then run the Eve algorithm with access to $\mathsf{PUF}'_{\mathcal{S}}$: $\mathcal{R}$ has access because it holds $\mathsf{PUF}'_{\mathcal{S}}$, and $\mathcal{S}$ has access because it knows the code in $\mathsf{PUF}'_{\mathcal{S}}$. An analogous statement holds for $\mathsf{PUF}'_{\mathcal{R}}$.

To handle the second set of queries above, we rely on the ability of $\mathcal{S}$ to create stateful PUFs. Specifically, we have $\mathcal{S}$ create $\mathsf{PUF}_{\mathcal{S}}$ in such a way that it records (in an undetectable fashion) all the queries that $\mathcal{R}$ makes to $\mathsf{PUF}_{\mathcal{S}}$, in such a way that $\mathcal{S}$ can later recover these queries once $\mathsf{PUF}_{\mathcal{S}}$ is back in its possession. (This is easy to do by hardcoding in the PUF a secret challenge, chosen in advance by $\mathcal{S}$, to which the PUF responds with the set of all queries made to the PUF.) So, at the end of the protocol, it is trivial for $\mathcal{S}$ to learn all the queries that $\mathcal{R}$ made to $\mathsf{PUF}_{\mathcal{S}}$. Of course, $\mathcal{R}$ knows exactly the set of queries it made to $\mathsf{PUF}_{\mathcal{S}}$ throughout the course of the protocol. Queries that $\mathcal{S}$ makes to $\mathsf{PUF}_{\mathcal{R}}$ are handled in a similar fashion.

To complete the proof, we then show that the set of intersection queries as defined above is enough for the analysis from [BM09; BM13] to go through. In order for our proof to go through, it is crucial to find intersection queries immediately after each message is sent, as opposed to waiting until the end of the protocol. This is necessary in order to ensure the *independence* of the views of $\mathcal{S}$ and $\mathcal{R}$. Therefore, we define a variant of the Eve algorithm which, after each protocol message is sent, makes queries to a particular set of PUFs, determined by the sets of PUFs currently held by each party. For example, if immediately after message i is sent $\mathcal{S}$ holds $\{\mathsf{PUF}_{\mathcal{S}}, \mathsf{PUF}'_{\mathcal{R}}\}$ and R holds $\{\mathsf{PUF}_{\mathcal{R}}, \mathsf{PUF}'_{\mathcal{S}}\}$, then our Eve variant will make queries only to $\mathsf{PUF}'_{\mathcal{R}}$ and $\mathsf{PUF}'_{\mathcal{S}}$.

2.2.2 Proof Details

Oblivious transfer. Oblivious transfer (OT) is a protocol between a sender $\mathcal{S}$ with input bits (s_0, s_1) and a receiver $\mathcal{R}$ with input bit b. Informally, the receiver wishes to retrieve s_b from $\mathcal{S}$ in such a way that (1) $\mathcal{S}$ does not "learn" anything about $\mathcal{R}$'s choice and (2) $\mathcal{R}$ learns nothing about s_{1-b}.

We note that our impossibility holds even for protocols that do not enjoy perfect correctness, i.e., it holds for protocols where correctness holds (over choice of inputs, randomness, and PUFs) with probability $1 - 1/\mathsf{poly}(n)$.

Protocols based on PUFs. We consider a candidate PUF-based OT protocol Π with ℓ rounds that has 2ℓ *passes* and where in each *pass* a party sends a message. We assume w.l.o.g. that $\mathcal{S}$ sends the first message of the protocol and $\mathcal{R}$ sends the final message. Let $z = z(n)$ be the total number of PUFs used in protocol Π with security parameter n. We model the set of all PUFs $\{\mathsf{PUF}_1, \ldots, \mathsf{PUF}_z\}$ utilized by Π as a single random oracle. W.l.o.g. we assume that each query q to a PUF has the form $q = (j, q')$, where j denotes the identity of the PUF being queried, and q' denotes the actual query to this PUF. [1] Note that when using this ideal PUF, responses to unique queries $q = (j, q')$ are independent and uniform. We further assume w.l.o.g. that a party will only send a PUF back and forth along with some message m_i of the protocol Π. In particular, we denote by S^i_{back} the set of indices $j \in [z]$ such that PUF_j is sent by $\mathcal{S}$ (resp. $\mathcal{R}$) to $\mathcal{R}$ (resp. $\mathcal{S}$) immediately after message m_i of Π is sent, and PUF_j was created by $\mathcal{R}$ (resp. $\mathcal{S}$). We define S^i_{PUF} to be the set of indices j such that either:

- PUF_j is held by $\mathcal{S}$ immediately after message m_i is sent and PUF_j was created by $\mathcal{R}$.
- PUF_j is held by $\mathcal{R}$ immediately after message m_i is sent and PUF_j was created by $\mathcal{S}$.

Augmented transcripts. A full (augmented) transcript of protocol $\Pi = \langle \mathcal{S}, \mathcal{R} \rangle$ is denoted by $\widetilde{\mathsf{M}}$. The "augmented transcript" consists of the transcript $\mathsf{M} = m_1, \ldots, m_{2\ell}$ of protocol Π with a set ψ^i appended after each message m_i. If message m_i is sent by $\mathcal{S}$ (resp. $\mathcal{R}$), then ψ^i contains all queries made by $\mathcal{S}$ (resp. $\mathcal{R}$) up to this point in the protocol to all PUF_j, $j \in S^i_{\mathsf{back}}$. Specifically, $\mathsf{M} = \{m_1, \ldots, m_{2\ell}\}$ and $\widetilde{\mathsf{M}} = \{m_1 \| \psi^1, \ldots, m_{2\ell} \| \psi^{2\ell}\}$. We also define the set Ψ^i which is the union of the ψ^j sets for $j \le i$. Specifically, $\Psi^i = \psi^1 \cup \cdots \cup \psi^i$. Note that $\widetilde{\mathsf{M}}$ can be computed by both a malicious $\mathcal{S}$ and a malicious $\mathcal{R}$ participating in Π. Intuitively, this is because both malicious $\mathcal{S}$ and $\mathcal{R}$ can program each of their PUFs to record all queries made to it. The following claim formalizes the fact that malicious, stateful PUFs can be used to extract sets of queries made by the opposite party:

Claim 3. *Consider a PUF-based ℓ-round OT protocol, Π. By participating in an execution of Π while using maliciously constructed PUFs, we have that, for all odd $i \in [2\ell]$,* both *a malicious $\mathcal{S}$ and a malicious $\mathcal{R}$ can find the set of queries ψ^i made by $\mathcal{S}$ up to this point in the protocol to all PUF_j, $j \in S^i_{\mathsf{back}}$. The same claim holds for even $i \in [2\ell]$, with the roles of $\mathcal{S}, \mathcal{R}$ reversed.*

[1] Jumping ahead to the formal treatment of PUFs in Section 2.3.4, this means that the PUF family is instantiated using a single random oracle, $\mathsf{S}_{\mathsf{nor}}(1^n)$ simply chooses a random id, and $\mathsf{E}_{\mathsf{nor}}(1^n, \mathsf{id}, c)$ responds by querying the random oracle on id and c.

Proof. The malicious $\mathcal{R}$ will instantiate their own PUFs using a stateful circuit that uses a PUF gate to honestly evaluate the PUF, but stores the query in the updated state such that this record can later be retrieved by the creator of the PUF. This can for example be achieved by having the circuit simply output the full state when queried on some randomly chosen secret input. Note that such a circuit would be polynomial in size and therefore a valid circuit for a maliciously generated PUF. Since at the end of the i-th pass, for odd $i \in [2\ell]$, $\mathcal{R}$ holds PUF_j, $j \in S^i_{\mathsf{back}}$, we have that the malicious $\mathcal{R}$ can recover the ordered set of queries made to that PUF, and can therefore deduce the set of queries made by $\mathcal{S}$ to that PUF thus far. On the other hand, $\mathcal{S}$ knows the queries it made itself to PUF_j, $j \in S^i_{\mathsf{back}}$. An analogous argument holds for even $i \in [2\ell]$. □

Queries and views. By $\mathrm{V}^i_{\mathcal{S}}$ (resp. $\mathrm{V}^i_{\mathcal{R}}$) we denote the view of $\mathcal{S}$ (resp. $\mathcal{R}$) up to the end of round i. This includes $\mathcal{S}$'s (resp. $\mathcal{R}$'s) randomness $r_{\mathcal{S}}$ (resp. $r_{\mathcal{R}}$), exchanged messages M^i as well as oracle query-answer pairs known to $\mathcal{S}$ (resp. $\mathcal{R}$) so far. We use $\mathcal{Q}(\cdot)$ as an operator that extracts the set of queries from a set of query-answer pairs or a view.

Executions and distributions. A (full) *execution* of $\mathcal{S}, \mathcal{R}$, Eve in protocol Π can be described by a tuple $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ where H is a random function. We denote by $\mathcal{E}$ the distribution over (full) executions that is obtained by running the algorithms for $\mathcal{S}, \mathcal{R}$, Eve with uniformly chosen random tapes and a sampled oracle H. For a sequence of i (augmented) messages $\widetilde{\mathsf{M}}^i = [\widetilde{m}_1, \ldots, \widetilde{m}_i]$ and a set of query-answer pairs P, by $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ we denote the joint distribution over the views $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}})$ of $\mathcal{S}$ and $\mathcal{R}$ in their (partial) execution of Π up to the point in the system in which the i-th message is sent (by $\mathcal{S}$ or $\mathcal{R}$) conditioned on: The transcript of messages in the first i passes equals $\widetilde{\mathsf{M}}^i$ and $H(j, q') = a$ for all $((j, q'), a) \in \mathsf{P}$ made to H (recall that a query (j, q') to H corresponds to a query q' made to PUF_j). For $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ such that $\Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i, \mathsf{P})\right] > 0$, the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ can be sampled by first sampling $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$ uniformly at random conditioned on being consistent with $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ and then deriving $\mathcal{S}$ and $\mathcal{R}$ views $\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}$ from the sampled $(r_{\mathcal{S}}, r_{\mathcal{R}}, H)$.

For $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ such that $\Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i, \mathsf{P})\right] > 0$, the event $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is defined over the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ and holds if and only if $\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \cap \mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \subseteq \mathsf{P}^+$, where $\mathsf{P}^+ = \mathsf{P} \cup \Psi^i$. For $\Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i, \mathsf{P})\right] > 0$ we define the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ to be the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ conditioned on $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$. For complete transcripts $\widetilde{\mathsf{M}}$, the distributions $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$ and $\mathcal{GV}(\widetilde{\mathsf{M}}, \mathsf{P})$ are defined similarly.

Transforming the protocol. We begin by transforming the OT protocol Π into one that has the following properties:

Semi-normal form: We define a semi-normal form for OT protocols, following [BM09; BM13]. A protocol is in semi-normal form if it fulfills the following two properties:

(1) $\mathcal{S}$ and $\mathcal{R}$ ask at most one query in each protocol round, and

(2) the receiver of the last message uses this message to compute its output and it does not query the oracle.

We start by converting our OT protocol Π into its semi-normal version. Note that any attack on the semi-normal version of Π can be translated into an attack on the original Π that makes the *same* number of queries [BM09; BM13]. Thus, in the following we present our attacks and analysis w.r.t. the semi-normal version of Π.

Using t-wise independent functions: Instead of instantiating the PUFs such that they evaluate an honest PUF gate, a malicious $\mathcal{R}$ (resp. $\mathcal{S}$) will create malicious stateful PUFs which evaluate a randomly chosen t-wise independent hash functions. We define the distribution $\mathcal{V}^t(\widetilde{\mathsf{M}},\mathsf{P})$ exactly like $\mathcal{V}(\widetilde{\mathsf{M}},\mathsf{P})$ except some subset of PUFs is instantiated with t-wise independent hash functions for some $t = \mathsf{poly}(m/\epsilon)$, instead of with random oracles. Since we choose t such that the malicious sender and honest receiver (resp., malicious receiver and honest sender) make a total of at most t queries to all PUFs then: For every setting of random variables $(\widetilde{\mathsf{M}}^i,\mathsf{P}^i)$, the distributions $\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P}^i)$ and $\mathcal{V}^t(\widetilde{\mathsf{M}}^i,\mathsf{P}^i)$ are identical. Thus, from now on, even when $\mathcal{R}$ or $\mathcal{S}$ are malicious (and create t-wise independent PUFs), we consider only the distribution $\mathcal{V}(\widetilde{\mathsf{M}},\mathsf{P})$.

Random inputs: In the last step we change the protocol such that both sender and receiver choose their input(s) uniformly at random. Thus, in the following, we consider execution of $\Pi = \langle \mathcal{S}(1^n), \mathcal{R}(1^n)\rangle$ where the parties use their random tapes to choose their inputs.

The Eve algorithm. Recall that we have converted the protocol Π into semi-normal form. We now present the attacking algorithm, Eve, which will be run by both the malicious sender ($\mathcal{S}'$) and malicious receiver ($\mathcal{R}'$) defined later:

Construction 1. *Let $\epsilon < 1/100$ be an input parameter. After each message m_i is sent,* Eve *generates the augmented transcript $\widetilde{\mathsf{M}}^i$ (note that by Claim 3, $\widetilde{\mathsf{M}}^i$ can always be reconstructed by* Eve*, since* Eve *is launched by either the malicious $\mathcal{S}$ or $\mathcal{R}$). Given $\widetilde{\mathsf{M}}^i$,* Eve *attacks the ℓ-round two-party protocol $\Pi = \langle \mathcal{S},\mathcal{R}\rangle$ as follows. During the attack* Eve *updates a set* P *of oracle query-answer pairs as follows: Suppose $\mathcal{S}$ (alternatively $\mathcal{R}$) sends the i-th message in $\widetilde{\mathsf{M}}^i$ which is equal to $\widetilde{m}_i = m_i \| \psi^i$. For $i \in [2\ell]$,* Eve *does the following: As long as the total number of queries made by* Eve *is less than $t - 2m$ and there is a query $q = (j,q') \notin \mathsf{P}^+$, where $\mathsf{P}^+ := \Psi^i \cup \mathsf{P}$, such that one of the following holds:*

$$\Pr_{(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\leftarrow\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[q' \in \mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \wedge j \in S^i_{\mathsf{PUF}}\right] \geq \frac{\epsilon^2}{100m}$$

or

$$\Pr_{(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\leftarrow\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[q' \in \mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \wedge j \in S^i_{\mathsf{PUF}}\right] \geq \frac{\epsilon^2}{100m}.$$

Eve *queries the lexicographically first such $q = (j,q')$ to H and adds $(q,H(q))$ to* P.

2.2.3 Analysis of the Eve Algorithm

We summarize some properties of the Eve algorithm that can be verified by inspection:

Symmetry of* Eve*: Both $\mathcal{S}$ and $\mathcal{R}$ can run the Eve algorithm, making the same set of queries P to the PUFs. In particular, at the point where message m_i is sent, a party requires only the augmented transcript $\widetilde{\mathsf{M}}^i$ and oracle access to PUF_j for $j \in S^i_{\mathsf{PUF}}$. Note that for each $j \in S^i_{\mathsf{PUF}}$ a party either holds PUF_j (and so can query it directly) or created PUF_j dishonestly and thus knows the code of PUF_j (and so can simulate responses to queries to PUF_j).

Determinism of* Eve*: The Eve algorithm is deterministic and so for a fixed transcript $\widetilde{\mathsf{M}}$ and a fixed set of PUFs, both parties will recover the same set of queries when running Eve.

Number of queries: The number of queries made by Eve is at most $t - 2m$. Thus, since $\mathcal{S}$ and $\mathcal{R}$ each make at most m number of PUF queries, the total number of queries made by $\mathcal{S}$, $\mathcal{R}$ and Eve is at most $(t-2m)+2m = t$.

We will now prove the following Lemma about the Eve algorithm:

Lemma 4. *Let $\Pi = \langle \mathcal{S}, \mathcal{R} \rangle$ be a PUF-based OT protocol in which the sender and receiver each ask at most m queries total. Then we have the following properties of the* Eve *algorithm, with input parameter $\epsilon < 1/100$, described in Construction 1:*

With probability at least $1 - \epsilon$ over the randomness of $\mathcal{S}$, $\mathcal{R}$, and the randomness of the PUFs we have that the statistical distance between $\mathcal{V}_{\mathrm{V}_\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P})$ and $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$ is at most ϵ. Namely:

$$\mathsf{SD}\Big(\mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathrm{V}_\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}), \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})\Big) \leq \epsilon.$$

We first consider a modified Eve algorithm which makes an *unbounded number* of queries and analyze the properties of this algorithm. We then switch to the Eve algorithm described in Construction 1. Our exposition follows [BM09; BM13] closely.

Construction 2. *Let $\epsilon < 1/100$ be an input parameter. After each message m_i is sent,* Eve *generates the augmented transcript $\widetilde{\mathsf{M}}^i$ (note that by Claim 3, $\widetilde{\mathsf{M}}^i$ can always be reconstructed by* Eve*, since* Eve *is launched by either the malicious $\mathcal{S}$ or $\mathcal{R}$). Given $\widetilde{\mathsf{M}}^i$,* Eve *attacks the ℓ-round two-party protocol $\Pi = \langle \mathcal{S}, \mathcal{R} \rangle$ as follows. During the attack* Eve *updates a set* P *of oracle query-answer pairs as follows: Suppose $\mathcal{S}$ (alternatively $\mathcal{R}$) sends the i-th message in $\widetilde{\mathsf{M}}^i$ which is equal to $\widetilde{m}_i = m_i \| \psi^i$. For $i \in [2\ell]$,* Eve *does the following: As long as there is a query $q = (j, q') \notin \mathsf{P}^+$, where $\mathsf{P}^+ := \Psi^i \cup \mathsf{P}$, such that one of the following holds:*

$$\Pr_{(\mathrm{V}^i_\mathcal{S}, \mathrm{V}^i_\mathcal{R}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}\Big[q' \in \mathcal{Q}(\mathrm{V}^i_\mathcal{S}) \wedge j \in S^i_{\mathsf{PUF}}\Big] \geq \frac{\epsilon^2}{100m}$$

or

$$\Pr_{(\mathrm{V}^i_\mathcal{S}, \mathrm{V}^i_\mathcal{R}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}\Big[q' \in \mathcal{Q}(\mathrm{V}^i_\mathcal{R}) \wedge j \in S^i_{\mathsf{PUF}}\Big] \geq \frac{\epsilon^2}{100m}$$

Eve *queries the lexicographically first such $q = (j, q')$ to H and adds $(q, H(q))$ to* P.

We prove the following:

Lemma 5. *Let $\Pi = \langle \mathcal{S}, \mathcal{R} \rangle$ be a PUF-based OT protocol in which the sender and receiver each ask at most m queries total. Then we have the following properties of the* Eve *algorithm, with input parameter $\epsilon < 1/100$, described in Construction 2:*

1. $\mathsf{poly}(m/\epsilon)$*-Efficiency:* Eve *is deterministic and, over the randomness of the PUFs and $\mathcal{S}$ and $\mathcal{R}$'s private randomness, the expected number of* Eve *queries is at most* $\mathsf{poly}(m/\epsilon)$.

2. $(1 - \epsilon/2)$*-Security: With probability at least $1 - \epsilon/2$ over the randomness of $\mathcal{S}$, $\mathcal{R}$, and the randomness of the PUFs we have that the statistical distance between $\mathcal{V}_{\mathrm{V}_\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P})$ and $\mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})$ is at most $\epsilon/2$. Namely:*

$$\mathsf{SD}\Big(\mathcal{V}_{\mathrm{V}_\mathcal{R}}(\widetilde{\mathsf{M}}, \mathsf{P}) \times \mathcal{V}_{\mathrm{V}_\mathcal{S}}(\widetilde{\mathsf{M}}, \mathsf{P}), \mathcal{V}(\widetilde{\mathsf{M}}, \mathsf{P})\Big) \leq \epsilon/2.$$

Preliminaries Let $\mathbf{H}$ be a random oracle, i.e., uniformly distributed over the set of all random functions $H : \{0,1\}^* \to \{0,1\}^*$. For any partial function F with domain D we denote by $\Pr_{\mathbf{H}}[F]$ the probability that the random oracle $\mathbf{H}$ is consistent with F.

Lemma 6. *For consistent finite partial functions F_1, F_2 and random oracle $\mathbf{H}$ it holds that*

$$\Pr_{\mathbf{H}}[F_1 \cup F_2] = \frac{\Pr_{\mathbf{H}}[F_1] \cdot \Pr_{\mathbf{H}}[F_2]}{\Pr_{\mathbf{H}}[F_1 \cap F_2]}.$$

Events over $\mathcal{E}$. Event Fail holds if and only if at *some* point during the execution of the system, $\mathcal{S}$ or $\mathcal{R}$ asks a query q, which was asked by the other party, but is not already contained in $\mathsf{P}^+ := \mathsf{P} \cup \Psi^i$. If the first query that makes Fail happen is $\mathcal{R}$'s j-th query we say the event RFail_j has happened, and if it is $\mathcal{S}$'s j-th query we say that the event SFail_j has happened.

Analysis of Modified Attack. Let $\epsilon_1 := \epsilon^2/100$. We prove the following lemma:

Lemma 7. *For* every $(\mathrm{V}^i_{\mathcal{R}}, \widetilde{\mathsf{M}}^i, \mathsf{P})$ *sampled by executing the system it holds that*

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[\mathsf{RFail}_i \mid \mathrm{V}^i_{\mathcal{R}}\right] \leq \frac{3\epsilon_1}{2m}.$$

A symmetric statement holds for $\mathcal{S}$.

The Graph Characterization As in [BM09; BM13], we first present a graph characterization for the distribution $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ and then use it to prove Lemma 7.

Lemma 8 (Graph Characterization of $\mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$)**.** *Let $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ be, in order, the augmented partial transcript and the set of oracle query-answer pairs known to* Eve *by the end of the round in which the last message in $\widetilde{\mathsf{M}}^i$ is sent, and $\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})\right] > 0$. For every such $(\widetilde{\mathsf{M}}^i, \mathsf{P})$, there is a bipartite graph G (depending on $(\widetilde{\mathsf{M}}^i, \mathsf{P})$) with vertices $(\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}})$ and edges E such that:*

1. *Every vertex u in $\mathcal{U}_{\mathcal{S}}$ has a corresponding view $\mathcal{S}_u$ for $\mathcal{S}$ and a set $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u) \setminus \mathsf{P}^+$, where $\mathsf{P}^+ := \mathsf{P} \cup \Psi^i$. The same holds for vertices in $\mathcal{U}_{\mathcal{R}}$ by changing the role of $\mathcal{S}$ and $\mathcal{R}$.*
2. *There is an edge between $u \in \mathcal{U}_{\mathcal{S}}$ and $v \in \mathcal{U}_{\mathcal{R}}$ if and only if $\mathcal{Q}_u \cap \mathcal{Q}_v = \emptyset$.*
3. *Every vertex is connected to at least $(1 - 2\epsilon_1)$ fraction of vertices on the other side.*
4. *The distribution $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is identical to sampling a random edge $(u, v) \leftarrow E$ and taking $(\mathcal{S}_u, \mathcal{R}_v)$ (i.e. the views corresponding to u and v).*

We now proceed to prove Lemma 7, given the graph characterization.

Proof. Let $\mathrm{V}^i_{\mathcal{R}}, \widetilde{\mathsf{M}}^i, \mathsf{P}$ be as in Lemma 7 and let $q = (j, q')$ be $\mathcal{R}$'s ℓ-th query which is going to be asked after the last message $\widetilde{m}_i$ in $\widetilde{\mathsf{M}}^i$ is sent to $\mathcal{R}$. By Lemma 8, the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ conditioned on getting $\mathrm{V}^i_{\mathcal{R}}$ as $\mathcal{R}$'s view is the same as uniformly sampling a random edge $(u, v) \leftarrow E$ in the graph G of Lemma 8 conditioned on $\mathcal{R}_v = \mathrm{V}^i_{\mathcal{R}}$. We prove Lemma 7 even conditioned on choosing any vertex v such that $\mathcal{R}_v = \mathrm{V}^i_{\mathcal{R}}$. For such fixed v, the distribution of $\mathcal{S}$'s view $\mathcal{S}_v$, when we choose a random edge (u, v') conditioned on $v = v'$ is the same as choosing

a random neighbor $u \leftarrow N(v)$ of the node v and then selecting $\mathcal{S}$'s view $\mathcal{S}_u$ corresponding to the node u. Let $S = \{u \in \mathcal{U}_{\mathcal{S}} \mid q \in \mathcal{Q}_u\}$. Note that if $q = (j, q')$ is such that $j \notin S^i_{\mathsf{PUF}}$ then we have that

$$\Pr_{u \leftarrow N(v)}[q \in \mathcal{Q}_u] = 0.$$

This is because $\mathcal{R}$ can only query a PUF that it holds at the point right after message $\widetilde{m}_i$ is sent. However, if PUF_j is such that $\mathcal{R}$ currently holds the PUF and $\mathsf{PUF}_j \notin S^i_{\mathsf{PUF}}$, then PUF_j must have been created by $\mathcal{R}$. Thus, by definition of the augmented transcript $\widetilde{\mathsf{M}}^i$, all queries made by $\mathcal{S}$ to PUF_j up to this point in the protocol are included in $\mathcal{S}^i \subseteq \mathsf{P}^+$ and thus cannot be in $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u) \setminus \mathsf{P}^+$.

Thus, we focus our attention on queries $q = (j, q')$ such that $j \in \mathcal{S}^i$. We have that

$$\Pr_{u \leftarrow N(v)}[q \in \mathcal{Q}_u] \leq \frac{|S|}{d(v)} \leq \frac{|S|}{(1-2\epsilon_1)\cdot \mathcal{U}_{\mathcal{S}}} \leq \frac{|S| \cdot |\mathcal{U}_{\mathcal{R}}|}{(1-2\epsilon_1)\cdot |E|} \leq \frac{\sum_{u \in S} d(u)}{(1-2\epsilon_1)^2 \cdot |E|} \leq \frac{\epsilon_1}{(1-2\epsilon_1)^2 \cdot m} \leq \frac{3\epsilon_1}{2m}$$

The second and fourth inequalities are due to the degree lower bounds of Item 3 in Lemma 8. The third inequality is because $|E| \leq |\mathcal{U}_{\mathcal{S}}| \cdot |\mathcal{U}_{\mathcal{R}}|$. The fifth inequality is because of the definition of the attacker Eve who asks ϵ_1/m heavy queries, for queries q of the form (j, q') where $j \in \mathcal{S}^i$, for $\mathcal{S}$'s view when sampled from $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$, as long as such queries exist. Namely, when we choose a random edge $(u, v) \leftarrow E$ (which by Lemma 8 is the same as sampling $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}) \leftarrow \mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$), it holds that $u \in S$ with probability $\sum_{u \in S} d(u)/|E|$. But for all $u \in S$ it holds that $q \in \mathcal{Q}_u$, and so if $\sum_{u \in S} d(u)/|E| > \epsilon_1/m$ the query q should have been learned by Eve already and by Item 2 of Lemma 8, q could not be in any set $\mathcal{Q}_u$. The sixth inequality is because we are assuming $\epsilon_1 < \epsilon < 1/100$. □

Next, we turn our attention to proving Lemma 8. In order to prove this Lemma, we present a product characterization of the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$.

Lemma 9 (Product Characterization)**.** *For any $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ there exists a distribution $\mathbf{S}$ (resp. $\mathbf{R}$) over $\mathcal{S}$'s (resp. $\mathcal{R}$'s) views such that the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is identical to the product distribution $(\mathbf{S} \times \mathbf{R})$ conditioned on the event $\mathsf{Good}(\widetilde{\mathsf{M}}^i, \mathsf{P})$. Namely,*

$$\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P}) \equiv ((\mathbf{S} \times \mathbf{R}) \mid \mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+)$$

Proof. Suppose $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}}) \leftarrow \mathcal{V}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is such that $\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \cap \mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \subseteq \mathsf{P}^+$, where $\mathsf{P}^+ = \mathsf{P} \cup \Psi^i$. Note that the set Ψ^i can be derived from $\widetilde{\mathsf{M}}^i$ only. For such $(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}})$ we will show that

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}})\right] = \alpha(\widetilde{\mathsf{M}}^i, \mathsf{P}) \cdot \alpha_{\mathcal{S}} \cdot \alpha_{\mathcal{R}},$$

where $\alpha(\widetilde{\mathsf{M}}^i, \mathsf{P})$ depends only on $(\widetilde{\mathsf{M}}^i, \mathsf{P})$, $\alpha_{\mathcal{S}}$ depends only on $\mathrm{V}^i_{\mathcal{S}}$, and $\alpha_{\mathcal{R}}$ depends only on $\mathrm{V}^i_{\mathcal{R}}$. This means that if we let $\mathbf{S}$ be the distribution over $\mathrm{Supp}\left(\mathrm{V}^i_{\mathcal{S}}\right)$ such that $\Pr_{\mathbf{S}}\left[\mathrm{V}^i_{\mathcal{S}}\right]$ is proportional to $\alpha_{\mathcal{S}}$ and let $\mathbf{R}$ be the distribution over $\mathrm{Supp}\left(\mathrm{V}^i_{\mathcal{R}}\right)$ such that $\Pr_{\mathbf{R}}\left[\mathrm{V}^i_{\mathcal{R}}\right]$ is proportional to $\alpha_{\mathcal{R}}$, then $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is proportional (and hence equal to) the distribution $((\mathbf{S} \times \mathbf{R}) \mid \mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+)$.

In the following we will show that $\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}}, \mathrm{V}^i_{\mathcal{R}})\right] = \alpha(\widetilde{\mathsf{M}}^i, \mathsf{P}) \cdot \alpha_{\mathcal{S}} \cdot \alpha_{\mathcal{R}}$. Since we are assuming $\mathcal{Q}(\mathbf{S}) \cap \mathcal{Q}(\mathbf{R}) \subseteq \mathsf{P}^+$ we have

$$\begin{aligned}\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\right] &= \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}}) \wedge \mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})\right] \\ &= \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[\mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]\cdot \Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\right].\end{aligned} \tag{2.1}$$

On the other hand, by definition of conditional probability we have

$$\Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\right] = \frac{\Pr_{\mathcal{E}}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})\right]}{\Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]}. \tag{2.2}$$

Therefore, by Equation 2.1 and Equation 2.2 we have

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\right] = \frac{\Pr_{\mathcal{E}}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})\right]}{\Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]\cdot \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[\mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]}. \tag{2.3}$$

The denominator of Equation 2.2 only depends on $(\widetilde{\mathsf{M}}^i,\mathsf{P})$ and so we can take $\beta(\widetilde{\mathsf{M}}^i,\mathsf{P}) = \Pr_{\mathcal{E}}\left[(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]\cdot \Pr_{\mathcal{V}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[\mathsf{Good}(\widetilde{\mathsf{M}}^i,\mathsf{P})\right]$. In the following we analyze the numerator.

Notation. For partial function F, we denote by $\Pr_{\mathcal{E}}[F] = \Pr_{\mathbf{H}}[F]$.

We claim that:

$$\Pr_{\mathcal{E}}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})\right] = \Pr[\mathbf{r}_{\mathcal{S}} = r_{\mathcal{S}}]\cdot\Pr[\mathbf{r}_{\mathcal{R}} = r_{\mathcal{R}}]\cdot\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\cup\mathsf{P}\right].$$

The reason is that the necessary and sufficient condition that $(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}},\widetilde{\mathsf{M}}^i,\mathsf{P})$ happens in the execution of the system is that when we sample a uniform $(r_{\mathcal{S}},r_{\mathcal{R}},H)$, $r_{\mathcal{S}}$ equals $\mathcal{S}$'s randomness, $r_{\mathcal{R}}$ equals $\mathcal{R}$'s randomness, and H is consistent with $\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\cup\mathsf{P}$. These conditions implicitly imply that $\mathcal{S}$ and $\mathcal{R}$ will indeed produce transcript $\widetilde{\mathsf{M}}^i$ as well.

Recall that $\mathsf{P}^+ = \mathsf{P}\cup\Psi^i$. Note that P^+ can be derived given $(\widetilde{\mathsf{M}}^i,\mathsf{P})$ only. Now by Lemma 6, the fact that $\Psi^i\subseteq\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})$ and the fact that $\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cap\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\subseteq\mathsf{P}\cup\Psi^i=\mathsf{P}^+$ we have that

$$\begin{aligned}\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\cup\mathsf{P}\right] &= \Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\cup\mathsf{P}^+\right] \\ &= \Pr_{\mathcal{E}}[\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}\left[\left(\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cup\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\right)\setminus(\mathsf{P}^+)\right] \\ &= \Pr_{\mathcal{E}}[\mathsf{P}^+]\cdot\frac{\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\setminus\mathsf{P}^+\right]\cdot\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\setminus\mathsf{P}^+\right]}{\Pr_{\mathcal{E}}\left[\left(\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\cap\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\right)\setminus\mathsf{P}^+\right]} \\ &= \Pr_{\mathcal{E}}[\mathsf{P}^+]\cdot\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\setminus\mathsf{P}^+\right]\cdot\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\setminus\mathsf{P}^+\right].\end{aligned}$$

Therefore, we get:

$$\Pr_{\mathcal{GV}(\widetilde{\mathsf{M}}^i,\mathsf{P})}\left[(\mathrm{V}^i_{\mathcal{S}},\mathrm{V}^i_{\mathcal{R}})\right] = \Pr[\mathbf{r}_{\mathcal{S}} = r_{\mathcal{S}}]\cdot\Pr[\mathbf{r}_{\mathcal{R}} = r_{\mathcal{R}}]\cdot\Pr_{\mathcal{E}}[\mathsf{P}^+]\cdot\frac{\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}})\setminus\mathsf{P}^+\right]\cdot\Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}})\setminus\mathsf{P}^+\right]}{\beta(\widetilde{\mathsf{M}}^i,\mathsf{P})}$$

and so we can take

$$\alpha_{\mathcal{S}} = \Pr[\mathbf{r}_{\mathcal{S}} = r_{\mathcal{S}}] \cdot \Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{S}}) \setminus \mathsf{P}^+\right], \quad \alpha_{\mathcal{R}} = \Pr[\mathbf{r}_{\mathcal{R}} = r_{\mathcal{R}}] \cdot \Pr_{\mathcal{E}}\left[\mathcal{Q}(\mathrm{V}^i_{\mathcal{R}}) \setminus \mathsf{P}^+\right], \quad \alpha(\widetilde{\mathsf{M}}^i, \mathsf{P}) = \frac{\Pr_{\mathcal{E}}[\mathsf{P}^+]}{\beta(\widetilde{\mathsf{M}}^i, \mathsf{P})}.$$

□

Using the product characterization from Lemma 9, we define the following graph. $\mathcal{GV}(\widetilde{M}^i, \mathsf{P})$ is a distribution over random edges of some bipartite graph G. More precisely, for fixed $(\widetilde{\mathsf{M}}^i, \mathsf{P})$ the bipartite graph $G = (\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}}, E)$ is defined as follows. Every node $u \in \mathcal{U}_{\mathcal{S}}$ will have a corresponding partial view $\mathcal{S}_u$ of $\mathcal{S}$ that is in the support of the distribution $\mathbf{S}$ from Lemma 9.

We let the number of nodes corresponding to a view $\mathrm{V}^i_{\mathcal{S}}$ be proportional to $\Pr_{\mathbf{S}}\left[\mathbf{S} = \mathrm{V}^i_{\mathcal{S}}\right]$, meaning that $\mathbf{S}$ corresponds to the uniform distribution over the left-side vertices $\mathcal{U}_{\mathcal{S}}$. Similarly, every node $v \in \mathcal{U}_{\mathcal{R}}$ will have a corresponding partial view $\mathcal{R}_v$ of $\mathcal{R}$ such that $\mathbf{R}$ corresponds to the uniform distribution over $\mathcal{U}_{\mathcal{R}}$.

We define $\mathcal{Q}_u = \mathcal{Q}(\mathcal{S}_u) \setminus \left(\mathsf{P} \cup \Psi^i\right) = \mathcal{Q}(\mathcal{S}_u) \setminus \mathsf{P}^+$ for $u \in \mathcal{U}_{\mathcal{S}}$ to be the set of queries *outside* P^+ that were asked by $\mathcal{S}$ in the view $\mathcal{S}_u$. We define $\mathcal{Q}_v = \mathcal{Q}(\mathcal{R}_v) \setminus \mathsf{P}^+$ similarly. We put an edge between the nodes u and v (denoted by $u \sim v$) in G if and only if $\mathcal{Q}_u \cap \mathcal{Q}_v = \emptyset$.

Lemma 9 implies that the distribution $\mathcal{GV}(\widetilde{\mathsf{M}}^i, \mathsf{P})$ is equal to the distribution obtained by letting (u, v) be a random edge of the graph G and choosing $(\mathcal{S}_u, \mathcal{R}_v)$. It turns out that the graph G is *dense* as formalized in the next lemma.

Lemma 10. *Let $G = (\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}})$ be the graph above. Then for every $u \in \mathcal{U}_{\mathcal{S}}$, $d(u) \geq |\mathcal{U}_{\mathcal{R}}| \cdot (1 - 2\epsilon_1)$ and for every $v \in \mathcal{U}_{\mathcal{R}}$, $d(v) \geq |\mathcal{U}_{\mathcal{S}}| \cdot (1 - 2\epsilon_1)$ where $d(w)$ is the degree of vertex w.*

Proof. We first show that for every $w \in \mathcal{U}_{\mathcal{S}}$, $\sum_{v \in \mathcal{U}_{\mathcal{R}}, w \nsim v} d(v) \leq \epsilon_1 \cdot |E|$. The reason is that when a random edge is chosen, the probability of each vertex v being chosen is $d(v)/|E|$ and if $\sum_{v \in \mathcal{U}_{\mathcal{R}}, w \nsim v} d(v)/|E| > \epsilon_1$ it means that $\Pr_{(u,v) \leftarrow E}[\mathcal{Q}_w \cap \mathcal{Q}_u \neq \emptyset] \geq \epsilon_1$. Moreover, note that $\mathcal{Q}_w \cap \mathcal{Q}_u$ can only contain queries of the form $q = (j, q')$ where $j \in S^i_{\mathsf{back}}$ since immediately after message i has been sent, for $j \notin S^i_{\mathsf{back}}$, P^+ contains all queries made by at least one of the parties to PUF_j. Hence, because $|\mathcal{Q}_w| \leq m$, by the pigeonhole principle there must exist $q = (j, q')$ where $j \in \mathsf{back}^i$ such that $\Pr_{(u,v) \leftarrow E}[q \in Q_v] \geq \epsilon_1/m$. But this is a contradiction, because if that holds, then q should have been in P by the definition of the attacker Eve of Construction 2, and hence it could not be in $\mathcal{Q}_w$. The same argument shows that for every $w \in \mathcal{U}_{\mathcal{R}}$, $\sum_{u \in \mathcal{U}_{\mathcal{S}}, w \nsim v} d(u) \leq \epsilon_1 \cdot |E|$. Thus for every vertex $w \in \mathcal{U}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{R}}$, $|E^{\nsim}(w)| \leq \epsilon_1 |E|$ where $E^{\nsim}(w)$ denotes the set of edges that do not contain any neighbor of w (i.e. $E^{\nsim}(w) = \{(u, v) \in E \mid u \nsim w \wedge w \nsim\}$). The following claim proved in [BM09; BM13] completes the proof of Lemma 10.

Claim 11. *For $\epsilon_1 \leq 1/2$, let $G = (\mathcal{U}_{\mathcal{S}}, \mathcal{U}_{\mathcal{R}}, E)$ be a nonempty bipartite graph where $|E^{\nsim}(w)| \leq \epsilon_1 \cdot |E|$ for all vertices $w \in \mathcal{U}_{\mathcal{S}} \cup \mathcal{U}_{\mathcal{R}}$. Then $d(u) \geq |\mathcal{U}_{\mathcal{R}}| \cdot (1 - 2\epsilon_1)$ for all $u \in \mathcal{U}_{\mathcal{S}}$ and $d(v) \geq |\mathcal{U}_{\mathcal{S}}| \cdot (1 - 2\epsilon_1)$ for all $v \in \mathcal{U}_{\mathcal{R}}$.*

□

Given Lemma 7, Lemma 5 holds via the same analysis as in [BM09; BM13].[2]

[2] In particular, given Lemma 7, the proofs of Lemmas 3.4 and 3.5 of [BM13] can be used, essentially unchanged, to complete the proof of our Lemma 5.

Analysis of the original attack. Note that the Eve of Construction 1 is the same as Eve defined in Construction 2 except it is modified to stop asking queries once Eve has made $t - 2m = \mathsf{poly}(m/\epsilon)$ number of queries. Since, by property (1) of Lemma 5 we have that the expected number of queries made by the Eve of Construction 2 is $\mathsf{poly}(m/\epsilon)$, for appropriate setting of t, Lemma 4 follows from property (2) of Lemma 5 and Markov's inequality. This completes the proof of Lemma 4.

Breaking oblivious transfer. Recall that we assume that the honest $\mathcal{S}$ chooses its inputs (s_0, s_1) at random and that the honest $\mathcal{R}$ chooses its input bit at random. Thus, we may consider an execution of OT protocol $\Pi = \langle \mathcal{S}(1^n), \mathcal{R}(1^n) \rangle$ where the parties use their random tapes to choose their inputs.

We now state an alternative version of Theorem 2:

Theorem 12. *Let $\Pi = \langle \mathcal{S}(1^n), \mathcal{R}(1^n) \rangle$ be a PUF-based OT-protocol in which the sender and receiver each ask at most m queries total to the set of $z = \mathsf{poly}(n)$ PUFs, $\{\mathsf{PUF}_1, \ldots, \mathsf{PUF}_z\}$. Then, at least one of the following must hold:*

1. *There exists an adversarial $\mathcal{S}$ that uses malicious, stateful PUFs to compute the choice bit of $\mathcal{R}$ with advantage $1/\mathsf{poly}(n)$ and makes $\mathsf{poly}(n)$ queries to the PUFs.*
2. *For constant ϵ, δ where $\epsilon < 1/100$ and $2\epsilon < \delta < 1/36$, there exists an adversarial $\mathcal{R}$ that uses malicious, stateful PUFs to correctly guess both secrets of $\mathcal{S}$ with probability $1 - 12\delta$ and makes $\mathsf{poly}(n)$ queries to the PUFs.*

By choosing constants ϵ, δ appropriately, we obtain the parameters of Theorem 2.

Proof. We begin with some notation. For a view $\mathrm{V}_{\mathcal{R}}$ (resp. $\mathrm{V}_{\mathcal{S}}$), we denote by $\mathsf{In}(\mathrm{V}_{\mathcal{R}})$ (resp. $\mathsf{In}(\mathrm{V}_{\mathcal{S}})$) the input of the corresponding party implicitly contained in its view. We denote by $\mathsf{Out}(\mathrm{V}_{\mathcal{R}})$ the output of $\mathcal{R}$ implicitly contained in its view. For a distribution $\mathcal{D}$ and random variables $X_1, \ldots, X_n$, we denote by $\mathcal{D}(X_1, \ldots, X_n)$ the distribution $\mathcal{D}$ conditioned on $X_1, \ldots, X_n$.

Let $p(\cdot)$ be some sufficiently large polynomial. We consider two cases.

Case 1: With probability $1/p(n)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$ generated by a run of $\tilde{\Pi}$ we have that either

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \leq 1/2 - \delta$$

$$\text{or} \quad \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] \leq 1/2 - \delta$$

holds, where $(s_0, s_1) = \mathsf{In}(\mathrm{V}_{\mathcal{S}})$.

Case 2: With probability $1 - 1/p(n)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$ generated by a run of $\tilde{\Pi}$ we have that both

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \geq 1/2 - \delta$$

$$\text{and} \quad \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] \geq 1/2 - \delta$$

hold, where $(s_0, s_1) = \mathsf{In}(\mathrm{V}_{\mathcal{S}})$.

Clearly, for any PUF-based OT protocol, either Case 1 or Case 2 must hold. We show that if Case 1 holds then a malicious sender may attack receiver privacy making $\mathsf{poly}(m/\epsilon)$ queries and succeeding with advantage $\delta/4p(n)$, and if Case 2 holds then a malicious receiver may attack sender privacy making $\mathsf{poly}(m/\epsilon)$ queries and succeeding with probability $1 - 10\delta$. This is sufficient to prove the theorem.

We next present and analyze the attacks on Receiver and Sender privacy.

2.2.3.1 Sender's attack (denoted $\mathcal{S}'$) on receiver privacy:

1. Participate in protocol Π where the PUFs constructed by $\mathcal{S}$ are instantiated with t-wise independent hash functions and maliciously constructed to record $\mathcal{R}$ queries.
2. Convert the resulting transcript M to the augmented transcript $\widetilde{\mathsf{M}}$ in an online fashion.
3. Run the Eve algorithm on augmented transcript $\widetilde{\mathsf{M}}$ in an online fashion to generate the set P.
4. Compute the probabilities

$$P_0 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0]$$
$$\text{and} \quad P_1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1].$$

5. If $P_0 \geq 1/2 + \delta/2$, output 0, if $P_1 \geq 1/2 + \delta/2$, output 1. Otherwise, output 0 or 1 with probability 1/2.

Success of the attack. Recall that if Case 1 occurs then with probability $1/p(n)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$ we have that either

$$\begin{aligned} &\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \leq \frac{1}{2} - \delta \\ \text{or} \quad &\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] \leq \frac{1}{2} - \delta \end{aligned} \tag{2.4}$$

where $(s_0, s_1) = \mathsf{In}(\mathrm{V}_{\mathcal{S}})$. Note that by assuming that the correctness of the OT protocol is at least $1 - \delta/4p(n)$, we have that

$$\Pr_{\mathcal{E}}[(\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_0) \vee (\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_1)] \leq \delta/4p(n).$$

Thus, by Markov's inequality, we have that with probability $1 - 1/2p(n)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$:

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[(\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_0) \vee (\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_1)] \leq \delta/2. \tag{2.5}$$

Combining Equation 2.4 and Equation 2.5 with the fact that for every $\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}}$ in the support of $\mathcal{E}$,

$$1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] + \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] + \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[(\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_0) \vee (\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) \neq s_1)],$$

we have that with probability $1/2p(n)$ over $(\widetilde{\mathsf{M}}, \mathsf{P}, \mathrm{V}_{\mathcal{S}})$ either:

$$P_0 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_0] \geq \frac{1}{2} + \delta/2$$

$$\text{or} \quad P_1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}}) = s_1] \geq \frac{1}{2} + \delta/2$$

We claim that $\mathcal{S}'$ achieves advantage of $\delta/4p(n)$ in guessing $\mathcal{R}$'s input bit (where probabilities are taken over all coins of $\mathcal{S}'$ and $\mathcal{R}$). To see this, note that when $P_0 \geq 1/2 + \delta/2$ or when $P_1 \geq 1/2 + \delta/2$, then $\mathcal{S}'$ guesses the $\mathcal{R}$'s input bit correctly with probability at least $1/2 + \delta/2$ over choice of PUFs and $\mathcal{R}$'s randomness. Thus, the sender's advantage is equal to:

$$\frac{1}{2}\left(1 - \frac{1}{2p(n)}\right) + \frac{1}{2p(n)}\left(\frac{1}{2} + \frac{\delta}{2}\right) - \frac{1}{2} = \frac{1}{2} + \frac{\delta}{4p(n)} - \frac{1}{2} = \frac{\delta}{4p(n)}.$$

This concludes the analysis of the sender's attack.

2.2.3.2 Receiver's attack (denoted $\mathcal{R}'$) on sender privacy:

1. Participate in protocol Π where the PUFs constructed by $\mathcal{R}$ are instantiated with t-wise independent hash functions and are maliciously constructed to record $\mathcal{S}$ queries.
2. Convert the resulting transcript M to the augmented transcript $\widetilde{\mathsf{M}}$ in an online fashion.
3. Run the Eve algorithm on augmented transcript $\widetilde{\mathsf{M}}$ in an online fashion to generate the set P.
4. Compute the probabilities

$$P_0 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0] \quad \text{and} \quad P_1 = \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1].$$

5. If $P_0 = 0$ or $P_1 = 0$ then output $\perp$ and terminate.
6. Otherwise, draw two views $\mathrm{V}_R(0)$ and $\mathrm{V}_R(1)$ from $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0)$ and $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1)$, respectively (where $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 0)$ denotes the distribution over the view of the receiver conditioned on $(\widetilde{\mathsf{M}}, \mathsf{P})$ and the receiver's input bit being equal to 0 and $\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}}, \mathsf{P}, \mathsf{In}(\mathrm{V}_{\mathcal{R}}) = 1)$ denotes the distribution over the view of the receiver conditioned on $(\widetilde{\mathsf{M}}, \mathsf{P})$ and the receiver's input bit being equal to 1).
7. Output $s'_0 = \mathsf{Out}(\mathrm{V}_R(0)), s'_1 = \mathsf{Out}(\mathrm{V}_R(1))$.

Success of the attack. We now proceed to analyze the success probability of the receiver's attack given Lemma 4. Recall that if Case 2 occurs then with probability $1-1/p(n)$ over $(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})$ we have that both:

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_0] \geq 1/2-\delta$$

$$\text{and} \quad \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_1] \geq 1/2-\delta$$

where $(s_0,s_1)=\mathsf{In}(\mathrm{V}_{\mathcal{S}})$.

This immediately implies that

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=0] \leq 1/2+\delta \text{ and } \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=1] \leq 1/2+\delta. \tag{2.6}$$

By Lemma 4 we have that with probability $1-\epsilon$ over $(\widetilde{\mathsf{M}},\mathsf{P})$

$$\mathsf{SD}\left(\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})\times\mathcal{V}_{\mathrm{V}_{\mathcal{S}}}(\widetilde{\mathsf{M}},\mathsf{P}),\mathcal{V}(\widetilde{\mathsf{M}},\mathsf{P})\right)\leq\epsilon. \tag{2.7}$$

Note that for any fixed $(\widetilde{\mathsf{M}},\mathsf{P})$,

$$\mathsf{SD}\left(\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})\times\mathcal{V}_{\mathrm{V}_{\mathcal{S}}}(\widetilde{\mathsf{M}},\mathsf{P}),\mathcal{V}(\widetilde{\mathsf{M}},\mathsf{P})\right)= \mathop{\mathbb{E}}_{\mathrm{V}_{\mathcal{S}}\sim\mathcal{V}_{\mathrm{V}_{\mathcal{S}}}(\widetilde{\mathsf{M}},\mathsf{P})}\left[\mathsf{SD}\left(\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P}),\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})\right)\right]. \tag{2.8}$$

Thus, using Markov's inequality and combining Equation 2.7 and Equation 2.8 we have that with probability $1-\sqrt{\epsilon}-\epsilon>1-2\sqrt{\epsilon}$ over $(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})$

$$\mathsf{SD}\left(\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P}),\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})\right)\leq\sqrt{\epsilon}.$$

Given the above, and since we assume $\delta\geq 2\sqrt{\epsilon}\geq 1/p(n)$ we have that with probability $1-2\delta$ over $(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})$ both

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=0 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_0] \geq \frac{1}{2}-2\delta$$

$$\text{and} \quad \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P})}[\mathsf{In}(\mathrm{V}_{\mathcal{R}})=1 \wedge \mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_1] \geq \frac{1}{2}-2\delta.$$

We denote the event that (over choice of $(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})$) both quantities above are at least $1/2-2\delta$ by EV. We must now analyze the probability that $s_0'=\mathsf{Out}(\mathrm{V}_R(0))$ and $s_1'=\mathsf{Out}(\mathrm{V}_R(1))$ are the correct input bits of the sender. First, note that event EV occurs with probability at least $1-2\delta$. Moreover, it is not hard to see that when event EV occurs for some fixed $(\widetilde{\mathsf{M}},\mathsf{P},\mathrm{V}_{\mathcal{S}})$, $\mathcal{R}'$ does not abort and moreover, due to Equation 2.6 we have that

$$\Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathsf{In}(\mathrm{V}_{\mathcal{R}})=0)}[\mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_0] \geq \frac{1/2-2\delta}{1/2+2\delta} \geq 1-5\delta$$

$$\text{and} \quad \Pr_{\mathcal{V}_{\mathrm{V}_{\mathcal{R}}}(\widetilde{\mathsf{M}},\mathsf{P},\mathsf{In}(\mathrm{V}_{\mathcal{R}})=1)}[\mathsf{Out}(\mathrm{V}_{\mathcal{R}})=s_1] \geq \frac{1/2-2\delta}{1/2+2\delta} \geq 1-5\delta.$$

Thus, conditioned on EV ocurring, the probability that the sampled $V_R(0)$ (resp. $V_R(1)$) is such that $\mathsf{Out}(V_R(0))$ (resp. $\mathsf{Out}(V_R(1))$) is incorrect is at most 5δ. Finally, by a union bound, we have that with probability at least $1 - 12\delta$ (over all coins of $\mathcal{S}$ and $\mathcal{R}'$) both the sampled views of $\mathcal{R}'$ output the correct s'_0, s'_1. This concludes the analysis of the receiver's attack. □

2.3 Formalizing Physically Uncloneable Functions

As mentioned before, in reality the ideal PUFs used for our impossibility result do not exist. In reality a physically uncloneable function (PUF) is a physical device with "random" behavior introduced through uncontrollable manufacturing variations during their fabrication. When a PUF is queried with a stimulus (i.e., a challenge), it produces a physical output (the response). The output of a PUF can be noisy; i.e., querying the PUF twice with the same challenge may yield distinct, but close, responses. Moreover, the response need not be uniform; it may instead only have high min-entropy. While we could ignore these inconvenient details for the negative result, because using an ideal version of PUFs makes the negative result stronger, this is not true for a positive result. For the positive result to be meaningful, we need to make sure, that the definition of PUFs used as a building block is instantiable. This means that we will have to formally define malicious PUFs in the UC-model explicitely allowing for noisiness. Prior work has shown that for almost all applications, by using fuzzy extractors, one can eliminate the noisiness of a PUF and make its output effectively uniform.

Formally, a PUF family is defined by two algorithms S and Eval. The index-sampling algorithm S, which corresponds to the PUF-fabrication process, takes as input the security parameter 1^n and returns as output an index id. The evaluation algorithm Eval takes as input an index id and a challenge[3] c, and generates as output the corresponding response r.

We do not require that S or Eval can be evaluated efficiently. In fact, these are meant to represent *physical* processes that generate a physical object and measure this object's behavior under various conditions. The index id is simply a formal placeholder that refers to a well-defined physical object; it does not in itself represent any meaningful information about how this object works.

Following [BFS+11], we define the two main security properties of PUFs: *unpredictability* and *uncloneability*. As noted earlier, for simplicity we consider only a strong form of unpredictability where the output of the PUF is uniform. Intuitively, uncloneability means that only one party can evaluate a PUF at a time. This is formally modeled using an ideal functionality, $\mathcal{F}_{\mathsf{PUF}}$, that enforces this. Details of this ideal functionality are given in Section 2.3.4.

Finally, we also allow for the possibility of a maliciously generated PUF whose behavior does not necessarily correspond to $(\mathsf{S}, \mathsf{Eval})$ as described above. We consider two possibilities here: The first possibility is a *malicious-but-stateless PUF* that may use an $\mathsf{E}_{\mathtt{mal}}$ procedure of the adversary's choice in place of the honest algorithm Eval. Whenever a party in possession of this PUF evaluates it, it receives $\mathsf{E}_{\mathtt{mal}}(c)$ instead of $\mathsf{Eval}_{\mathsf{id}}(c)$. (As noted in prior work, care must be taken to ensure that the adversary cannot use $\mathsf{E}_{\mathtt{mal}}$ to perform arbitrary exponential-time computation; formally, we restrict $\mathsf{E}_{\mathtt{mal}}$ to be a polynomial-time algorithm with oracle access to $\mathsf{Eval}_{\mathsf{id}}$.) The second possibility is a *malicious-and-stateful PUF* that may use a *stateful* $\mathsf{E}_{\mathtt{mal}}$

[3] We assume the challenge space is just a set strings of a certain length. For some classes of PUFs, this is naturally satisfied (see [MV10]). For others, this can be achieved using appropriate encoding.

procedure of the adversary's choice in place of Eval. Again, $\mathsf{E}_{\mathtt{mal}}$ is limited to polynomial-time computation with oracle access to $\mathsf{Eval}_{\mathsf{id}}$.

To simplify notation throughout the rest of the paper, we write $\mathsf{PUF} \leftarrow \mathsf{S}(1^n)$ to denote the fabrication of a PUF, and then write $r := \mathsf{PUF}(c)$.

2.3.1 Security of PUFs

Many different security properties of PUFs have been suggested in the literature such as, e.g., unpredictability, uncloneability, bounded noise, uncorrelated outputs, one-wayness, and tamper-evidence. We refer the reader to [MV10; RSS09; BFS+11] for comprehensive discussions and overviews. Following [BFS+11], we stick to the two main security properties of PUFs: *uncloneability* and *unpredictability* (notice that unpredictability as defined here implies mild forms of uncloneability [BFS+11]).

Unpredictability. Loosely speaking, a PUF should be unpredictable in the sense that the PUF on input a challenge c has some significant amount of intrinsic entropy, even if the PUF has been measured before on several challenge values. The main tool to analyze this is (*conditional*) *min-entropy* which is a measurement for the min-entropy on a response value for a challenge c, when one has already measured the PUF on (not necessarily different) challenges $c_1, \ldots, c_\ell$ before. As discussed in [BFS+11] it is sufficient for our definition that a PUF has a certain *average* min-entropy, which we define as follows:

Definition 13 (Average Min-Entropy)**.** *The* average min-entropy *of* $\mathsf{PUF}(c)$ *conditioned on the measurements of challenges* $\mathcal{C} = (c_1, \ldots, c_\ell)$ *is defined by*

$$\tilde{\mathrm{H}}_\infty(\mathsf{PUF}(c)|\mathsf{PUF}(\mathcal{C})) := -\log\left(\mathop{\mathbb{E}}_{r_i:=\mathsf{PUF}(c_i)}\left[\max_r\left(\Pr\left[\mathsf{PUF}(c)=r|r_1=\mathsf{PUF}(c_1),\ldots,r_\ell=\mathsf{PUF}(c_\ell)\right]\right)\right]\right)$$

$$= -\log\left(\mathop{\mathbb{E}}_{r_i:=\mathsf{PUF}(c_i)}\left[2^{-\mathrm{H}_\infty(\mathsf{PUF}(c)|r_1=\mathsf{PUF}(c_1),\ldots,r_\ell=\mathsf{PUF}(c_\ell))}\right]\right)$$

where the probability is taken over the choice of id *from* $\mathcal{I}$ *and the choice of possible PUF responses on challenge* c*. The term* $\mathsf{PUF}(\mathcal{C})$ *denotes a sequence of random variables* $\mathsf{PUF}(c_1), \ldots, \mathsf{PUF}(c_\ell)$ *each corresponding to an evaluation of the PUF on challenge* c_k*. We set* $\tilde{\mathrm{H}}_\infty(\mathsf{PUF}(c)|\mathcal{C}) := \tilde{\mathrm{H}}_\infty(\mathsf{PUF}(c)|\mathsf{PUF}(\mathcal{C}))$*.*

Unpredictability is then defined as follows:

Definition 14 (Unpredictability)**.** *We call a* (rg, d_{noise})*-PUF family* $\mathcal{P} = (\mathsf{S}, \mathsf{Eval})$ *for security parameter* 1^n *is* $(d_{\mathsf{min}}(n), m(n))$-unpredictable *if for any* $c \in \{0,1\}^n$ *and any challenge list* $\mathcal{C} = (c_1, \ldots, c_\ell)$*, one has that, if for all* $1 \le k \le \ell$ *the Hamming distance satisfies* $\mathsf{D}_{\mathsf{ham}}(c, c_k) \ge d_{\mathsf{min}}(n)$*, then the average min-entropy satisfies* $\tilde{\mathrm{H}}_\infty(\mathsf{PUF}(c)|\mathsf{PUF}(\mathcal{C})) \ge m(n)$*. Such a PUF-family is called a* $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$*-PUF family.*

2.3.2 Applying Fuzzy Extractors to PUFs.

As mentioned before, the outputs of a PUF are noisy by nature. A natural way to handle these outputs is to apply a fuzzy extractors, as introduced by Dodis et al. [DRS04; DOR+08], to

convert any noisy, high-entropy measurements of PUFs into reproducible random values. Our definition of fuzzy extractors follows the one of [BFS+11] and we use the following notions: U_ℓ is the uniform distribution on ℓ-bit binary strings and a set $\mathcal{M}$ with a distance function $\mathsf{D} : \mathcal{M} \times \mathcal{M} \leftarrow \mathbb{R}^+ = [0, \infty)$ is called a metric space.

Definition 15 (Fuzzy Extractor)**.** *Let* D *be a distance function for metric space* $\mathcal{M}$*. A* (m, ℓ, t, ϵ)*-fuzzy extractor* FE *consists of two efficient randomized algorithms* $(\mathsf{Gen}, \mathsf{Rep})$*:*

Gen**:** *The algorithm* Gen *outputs on input* $w \in \mathcal{M}$ *a secret string* $st \in \{0,1\}^\ell$ *and a helper data string* $p \in \{0,1\}^*$*.*

Rep**:** *The algorithm* Rep *takes an element* $w' \in \mathcal{M}$ *and a helper data string* $p \in \{0,1\}^*$ *and outputs a string* st*.*

Correctness: *If* $\mathsf{D}(w, w') \leq t$ *and* $(st, p) \leftarrow_\$ \mathsf{Gen}(w)$*, then* $\mathsf{Rep}(w', p) = st$*.*

Security: *For any distribution* $\mathcal{W}$ *on the metric space* $\mathcal{M}$ *of min-entropy* m*, the first component of the random variable* (st, p)*, defined by drawing* w *according to* $\mathcal{W}$ *and then applying* Gen*, is distributed almost uniformly, even if* p *is observed, i.e.,* $\mathsf{SD}((st, p), (U_\ell, p)) \leq \epsilon$*.*

To combine PUFs with a fuzzy extractors one needs to define matching parameters such that the outputs of the composition of both primitives are almost uniformly distributed. Assume that we have a $(rg(n), d_{\mathsf{noise}}(n), d_{\mathsf{min}}(n), m(n))$-PUF family with d_{min} being in the order of $o(n/\log n)$ and let $\ell(n) := n$ be the length parameter for value st. By $\epsilon(n)$ we denote a negligible function and let $t(n) = d_{\mathsf{noise}}(n)$. For each n, let $(\mathsf{Gen}, \mathsf{Rep})$ be a $(m(n), \ell(n), t(n), \epsilon(n))$-fuzzy extractor. The metric space $\mathcal{M}$ is $\{0,1\}^{rg(n)}$ with Hamming distance $\mathsf{D}_{\mathsf{ham}}$.

Definition 16 (Matching Parameters)**.** *If a PUF and a fuzzy extractor* $\mathsf{FE} = (\mathsf{Gen}, \mathsf{Rep})$ *satisfy the above requirements, then they are said to have* matching parameters*.*

If a PUF and a fuzzy extractor have matching parameters, then the properties well-spread domain, extraction independence and response consistency follow.

Well-Spread Domain: For all polynomials $p(n)$ and all sets of challenges $c_1, ..., c_{p(n)}$, the probability of a random challenge to be within distance smaller d_{min} of any of the c_k is negligible.

Extraction Independence: For all challenges $c_1, ..., c_{p(n)}$, it holds that the PUF evaluation on a challenge c with $\mathsf{D}(c_k, c) > d_{\mathsf{min}}$ for all $1 \leq k \leq p(n)$ and subsequent application of Gen yields an almost uniform value st even for those who observe p.

Response Consistency: The fuzzy extractor helps to map two evaluations of the same PUF to the same random string, i.e., if PUF is measured on challenge c twice and returns r and r', then for $(st, p) \leftarrow_\$ \mathsf{Gen}(r)$, one has $st = \mathsf{Rep}(r', p)$.

2.3.3 Malicious PUFs in the UC Framework

Brzuska et al. [BFS+11] modeled PUFs in Canetti's universal composability framework [Can01]. Their ideal functionality considers only honestly generated PUFs and it is provided here for completeness. Subsequently, Ostrovsky et al. initiated the study of UC secure protocols in the

context of maliciously generated PUFs [OSV+12]. We first review the ideal functionality of honest PUFs, taken almost verbatim from [BFS+11].

The ideal functionality $\mathcal{F}_{\mathsf{HPUF}}$ is presented in Figure 2.1 and it allows the following operations: (1) a party $\mathcal{S}$ is allocated a PUF; (2) $\mathcal{S}$ can query the PUF; (3) $\mathcal{S}$ gives the PUF to another party $\mathcal{R}$ who can also query the device; (4) an adversary can query the PUF in transit [BFS+11].

The functionality $\mathcal{F}_{\mathsf{HPUF}}$ maintains a list $\mathcal{L}$ of tuples $(\mathtt{sid}, \mathsf{id}, \mathcal{S}, \tau)$ where $\mathtt{sid}$ is the (public) session identifier and id is the (internal) PUF-identifier, essentially describing the output distribution. Note that the PUF itself does not use $\mathtt{sid}$. The element $\tau \in \{\mathsf{trans}(\mathcal{R}), \mathsf{notrans}\}$ denotes whether the PUF is in transition to $\mathcal{R}$. For $\mathsf{trans}(\mathcal{R})$, indicating that the PUF is in transit to $\mathcal{R}$, the adversary is able to query the PUF. In turn, if it is set to $\mathsf{notrans}$ then only the possessing party can query the PUF.

The PUF functionality $\mathcal{F}_{\mathsf{HPUF}}$ is indexed by the PUF parameters $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$-PUF and gets the security parameter n in unary encoding as additional input. It is required to satisfy the bounded noise property for $d_{\mathsf{noise}}(n)$ and the unpredictability property for $(d_{\mathsf{min}}(n), m(n))$. This enforces that the outputs obey the basic entropic requirements of PUFs (analogously to the requirement for the random oracle functionality to produce random and independent outputs). We write $\mathcal{F}_{\mathsf{HPUF}}$ and $\mathcal{F}_{\mathsf{HPUF}}(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$ interchangeably.

Also note that our definition requires that a PUF is somehow certified. That is, the adversary cannot replace a PUF sent to an honest party by a fake token including some "software emulation"; the adversary can only measure the PUF when in transition. The receiver can verify the constitution and authenticity of the received hardware. Even though this certification is not explicitly modeled in the ideal functionality, it is easy to see that it is ensured since the receiver of the PUF can simply check if the $\mathtt{sid}$ it received is the correct one.

2.3.3.1 The Approach of Ostrovsky et al.

Ostrovsky et al. considered two different generalizations of the definition by Brzuska et al.. The first one models the case where the adversary is able to produce a fake PUF that behaves arbitrarily. The second definition allows the adversary to access the honestly generated PUF maliciously. In this work we are only interested in the setting where the PUF has been generated maliciously and will not discuss the second setting.

Their ideal functionality for maliciously generated PUFs is parametrized by two PUF families in order to handle honestly and maliciously generated PUFs: The honestly generated family is a pair $(\mathsf{S}_{\mathtt{nor}}, \mathsf{E}_{\mathtt{nor}})$ and the malicious one is $(\mathsf{S}_{\mathtt{mal}}, \mathsf{E}_{\mathtt{mal}})$. The malicious family can be considered as chosen by the attacker. Whenever a party P_i initializes a PUF, then it specifies whether it is an honest or a malicious PUF by sending $\mathtt{mode} \in \{\mathtt{nor}, \mathtt{mal}\}$ to the functionality $\mathcal{F}_{\mathsf{PUF}}$. The ideal functionality then initialises the appropriate PUF family and stores a tag $\mathtt{nor}$ or $\mathtt{mal}$ representing this family. Whenever the PUF is evaluated, the ideal functionality uses the evaluation algorithm that corresponds to the tag.

The handover procedure is identical to the original formulation of Brzuska et al. [BFS+11], where each PUF has a status flag $\tau \in \{\mathsf{trans}(\mathcal{R}), \mathsf{notrans}\}$ that indicates if a PUF is in transit or not. A PUF that is in transit can be queried by the adversary. Thus, whenever a party P_i sends a PUF to P_j, then the status flag is changed from $\mathsf{notrans}$ to trans and the attacker can evaluate the PUF. At some point, the attacker sends $\mathsf{ready}_{\mathsf{PUF}}$ to the ideal functionality to indicate that it is not querying the PUF anymore. The ideal functionality then hands the PUF over to P_j and

$\mathcal{F}_{\mathsf{HPUF}}$ is parameterized by the PUF family $\mathcal{P} = (\mathsf{S}_{\mathsf{nor}}, \mathsf{E}_{\mathsf{nor}})$ with parameters $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$, it also receives as initial input a security parameter 1^n and runs with parties $P_1, ..., P_n$ and adversary Sim.

- Whenever a party $\mathcal{S}$ sends $(\mathsf{init}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ to $\mathcal{F}_{\mathsf{HPUF}}$, $\mathcal{F}_{\mathsf{HPUF}}$ checks, if there exists a tuple $(\mathtt{sid}, *, *, *) \in \mathcal{L}$:
  - ⋆ If such a tuple exists return into waiting state.
  - ⋆ Else, draw $\mathsf{id} \leftarrow_{\$} \mathsf{S}_{\mathsf{nor}}(1^n)$ from the PUF-family, append $(\mathtt{sid}, \mathsf{id}, \mathcal{S}, \mathsf{notrans})$ to $\mathcal{L}$, and send $(\mathsf{initialized}_{\mathsf{PUF}}, \mathtt{sid})$ to $\mathcal{S}$.
- Whenever a party $\mathcal{S}$ sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, c)$ to $\mathcal{F}_{\mathsf{HPUF}}$, $\mathcal{F}_{\mathsf{HPUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \mathcal{S}, \mathsf{notrans}) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, run $r \leftarrow_{\$} \mathsf{E}_{\mathsf{nor}}(1^n, \mathsf{id}, c)$ and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to $\mathcal{S}$.
- Whenever a party $\mathcal{S}$ sends $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, \mathcal{R})$ to $\mathcal{F}_{\mathsf{HPUF}}$, $\mathcal{F}_{\mathsf{HPUF}}$ checks if there exists a tuple $(\mathtt{sid}, *, \mathcal{S}, \mathsf{notrans}) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, modify the tuple $(\mathtt{sid}, \mathsf{id}, \mathcal{S}, \mathsf{notrans})$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \perp, \mathsf{trans}(\mathcal{R}))$ and send $\mathsf{handover}_{\mathsf{PUF}}(\mathtt{sid}, \mathcal{S}, \mathcal{R})$ to Sim to indicate that a handover occurs between $\mathcal{S}$ and $\mathcal{R}$.
- Whenever Sim sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, \mathsf{Sim}, c)$ to $\mathcal{F}_{\mathsf{HPUF}}$, $\mathcal{F}_{\mathsf{HPUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \perp, \mathsf{trans}(*)) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, run $r \leftarrow_{\$} \mathsf{E}_{\mathsf{nor}}(1^n, \mathsf{id}, c)$ and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to Sim.
- Whenever Sim sends $(\mathsf{ready}_{\mathsf{PUF}}, \mathtt{sid}, \mathsf{Sim})$ to $\mathcal{F}_{\mathsf{HPUF}}$, $\mathcal{F}_{\mathsf{HPUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \perp, \mathsf{trans}(\mathcal{R})) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, modify the tuple $(\mathtt{sid}, \mathsf{id}, \perp, \mathsf{trans}(\mathcal{R}))$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \mathcal{S}, \mathsf{notrans})$, send $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ to $\mathcal{R}$, and store the tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$.
- Whenever Sim sends $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ on $\mathcal{F}_{\mathsf{HPUF}}$'s input tape, $\mathcal{F}_{\mathsf{HPUF}}$ checks if a tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ has been stored. If so, it sends this tuple to $\mathcal{S}$. Else, $\mathcal{F}_{\mathsf{HPUF}}$ returns into waiting state.

Figure 2.1: The ideal functionality $\mathcal{F}_{\mathsf{HPUF}}$ for honestly generated PUFs.

changes the status flag back to notrans. The party P_j may evaluate the PUF. Finally, when the attacker sends the message $\mathsf{received}_{\mathsf{PUF}}$ to the ideal functionality, then $\mathcal{F}_{\mathsf{PUF}}$ sends $\mathsf{received}_{\mathsf{PUF}}$ to P_i in order to notify P_i that the handover is over.

This model for maliciously generated PUFs was used as the basis of the previous version of this work [DFK+14] published at CRYPTO 2014, however, it has several problems. First of all, the fact that a malicious PUF can essentially contain a non-polynomial time algorithm presents a problem for any simulator in the security proof. Ostrovsky et al. try to circumvent this problem by requiring that a PUF family is *admissible* relative to the assumptions underlying the rest of the protocol. I.e., the problem should remain hard, even if the attacker is given access to the PUF family. However this does not seem to be enough. Consider the following example that illustrates the problem:

An attacker might embed an algorithm solving a *completely unrelated* hard problem into the

PUFs (e.g., the rest of the problem is based on DDH and the attacker embeds an RSA solver into the PUF) and simply probe the PUF before running the attack. If the PUF is able to solve the hard problem, it continues. Otherwise it aborts. This presents a problem for a polynomial time simulator, since it may have to simulate this super-polynomial time algorithm to effectively use the attacker. The admissibility requirement does not solve this, as the problem that is solved by the PUF is completely unrelated and therefore does not invalidate the assumptions underlying the rest of the protocol.

Another problem with the definition of Ostrovsky et al. is mainly counter-intuitive in the case where malicious PUFs can be stateful but becomes very pronounced in the case of stateless malicious PUFs: The only way for the attacker to instantiate a PUF, is to sample uniformly from the specified family. This presents a problem in several cases. An attacker might want to adaptively use several different kinds of malicious PUFs with different behavior, depending on the current protocol step. An attacker might want instantiate the PUFs as t-wise independent functions and use the fact that this allows them to evaluate the PUF, even if it is not in their possession. An attacker might want to fix the PUF to particular values on a fixed number of points but let it behave honestly everywhere else. In the stateful case, all of these and similar problems can be solved by giving the PUFs some kind of programming interface that allows the attacker to switch between different kinds of PUFs, program a function to evaluate, or program certain points of the PUF. While this seems like a very counter-intuitive approach, it does indeed work and gives the attacker a lot of power. As soon as we require PUFs to be stateless, however, the attacker loses any ability to change the behavior of the sampled malicious PUF and thus the power of an attacker degenerates much more severely than one would have hoped. In the next section, we therefore develop a new ideal functionality for maliciously generated PUFs that adresses these problems.

2.3.4 Maliciously Generated PUFs

The main change is, that malicious PUFs are no longer generated by drawing uniformly from an arbitrary attacker-specified family. Instead, to generate a malicious PUF, the attacker specifies a polynomial-size circuit that may include honest PUF gates. This gets rid of the possibility of a malicious PUF solving a hard problem, since the circuit must be of polynomial size and can, therefore, by definition not be used to solve any hard problems. It also makes specifying malicious PUFs much more intuitive, since it allows to directly specify the behavior, without having to use programming techniques. For the same reason, the power of an attacker also degenerates much more gracefully when restricted to stateless PUFs than it does in in the model of Ostrovsky et al. and all the examples from above can be realized in an intuitive way whether the PUFs are allowed to be stateless or not: If the attacker wants to specify different kinds of malicious PUFs at different points in the protocol, it may do so by simply specifying different circuits. If an attacker wants to instantiate the PUF with an efficiently computable function to allow evaluation even after handing the PUF over, it can do so by simply specifying said function as a circuit when instantiating the PUF. If an attacker wants to have an “almost honest” PUF that is fixed on only a few points, the honest PUF gates in the circuits easily allow to do this. The ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ is presented in Figure 2.2.

$\mathcal{F}_{\mathsf{PUF}}$ is parameterized by the PUF family $\mathcal{P} = (\mathsf{S}_{\mathrm{nor}}, \mathsf{E}_{\mathrm{nor}})$ with parameters $(rg, d_{\mathsf{noise}}, d_{\mathsf{min}}, m)$, it also receives as initial input a security parameter 1^n and runs with parties $P_1, ..., P_n$ and adversary Sim.

- Whenever a party $\mathcal{S}$ sends $(\mathsf{init}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, C, \mathsf{state})$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if there exists a tuple $(\mathtt{sid}, *, *, *, *, *, *) \in \mathcal{L}$:
  - ⋆ If such a tuple exists return into waiting state.
  - ⋆ Else, draw $\mathsf{id} \leftarrow_{\$} \mathsf{S}_{\mathrm{nor}}(1^n)$ from the PUF-family. If $C = \perp$ set $\mathtt{mode} = \mathtt{nor}$, otherwise set $\mathtt{mode} = \mathtt{mal}$. Append $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{state})$ to $\mathcal{L}$ and send $(\mathsf{initialized}_{\mathsf{PUF}}, \mathtt{sid})$ to $\mathcal{S}$.
- Whenever a party $\mathcal{S}$ sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, c)$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{state}) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, if $\mathtt{mode} = \mathtt{nor}$ run $r \leftarrow_{\$} \mathsf{E}_{\mathrm{nor}}(1^n, \mathsf{id}, c)$ and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to $\mathcal{S}$.
  - ⋆ Else, if $\mathtt{mode} = \mathtt{mal}$ evaluate $(\mathsf{state}', r) \leftarrow_{\$} C^{\mathsf{E}_{\mathrm{nor}}(1^n, \mathsf{id}, \cdot)}(\mathsf{state}, c)$, modify the tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mal}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{state})$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mal}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{state}')$, and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to $\mathcal{S}$.
- Whenever a party $\mathcal{S}$ sends $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, \mathcal{R})$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if there exists a tuple $(\mathtt{sid}, *, *, \mathcal{S}, \mathsf{notrans}, *, *) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, modify the tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{state})$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(\mathcal{R}), C, \mathsf{state})$ and send $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S}, \mathcal{R})$ to Sim to indicate that a $\mathsf{handover}_{\mathsf{PUF}}$ occurs between $\mathcal{S}$ and $\mathcal{R}$.
- Whenever Sim sends $(\mathsf{eval}_{\mathsf{PUF}}, \mathtt{sid}, \mathsf{Sim}, c)$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(*), C, \mathsf{state}) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, if $\mathtt{mode} = \mathtt{nor}$ run $r \leftarrow_{\$} \mathsf{E}_{\mathrm{nor}}(1^n, \mathsf{id}, c)$ and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to Sim.
  - ⋆ Else, if $\mathtt{mode} = \mathtt{mal}$ evaluate $(\mathsf{state}', r) \leftarrow_{\$} C^{\mathsf{E}_{\mathrm{nor}}(1^n, \mathsf{id}, \cdot)}(\mathsf{state}, c)$, modify the tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mal}, \perp, \mathsf{trans}(\mathcal{R}), C, \mathsf{state})$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mal}, \perp, \mathsf{trans}(\mathcal{R}), C, \mathsf{state}')$ and send $(\mathsf{response}_{\mathsf{PUF}}, \mathtt{sid}, c, r)$ to Sim.
- Whenever Sim sends $(\mathsf{ready}_{\mathsf{PUF}}, \mathtt{sid}, \mathsf{Sim})$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if there exists a tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(\mathcal{R}), *, *) \in \mathcal{L}$:
  - ⋆ If no such tuple exists return into waiting state.
  - ⋆ Else, modify the tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \perp, \mathsf{trans}(\mathcal{R}), C, \mathsf{state})$ to the updated tuple $(\mathtt{sid}, \mathsf{id}, \mathtt{mode}, \mathcal{S}, \mathsf{notrans}, C, \mathsf{trans})$, send $(\mathsf{handover}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ to $\mathcal{R}$, and store the tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$.
- Whenever Sim sends $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ to $\mathcal{F}_{\mathsf{PUF}}$, $\mathcal{F}_{\mathsf{PUF}}$ checks if a tuple $(\mathsf{received}_{\mathsf{PUF}}, \mathtt{sid}, \mathcal{S})$ has been stored. If so, it sends this tuple to $\mathcal{S}$. Else, $\mathcal{F}_{\mathsf{PUF}}$ returns into waiting state.

Figure 2.2: The ideal functionality $\mathcal{F}_{\mathsf{PUF}}$ for maliciously generated PUFs.

2.4 Feasibility Result for Malicious, Stateless PUFs

We show that universally composable two-party computation is possible if the adversary is limited to creating *stateless* malicious PUFs. The core of our result is a construction of an unconditionally secure, universally composable, oblivious-transfer protocol in this model; we describe the protocol here, and prove it secure. In Section 2.4.2 we briefly discuss how the oblivious-transfer protocol can be used to obtain the claimed result.

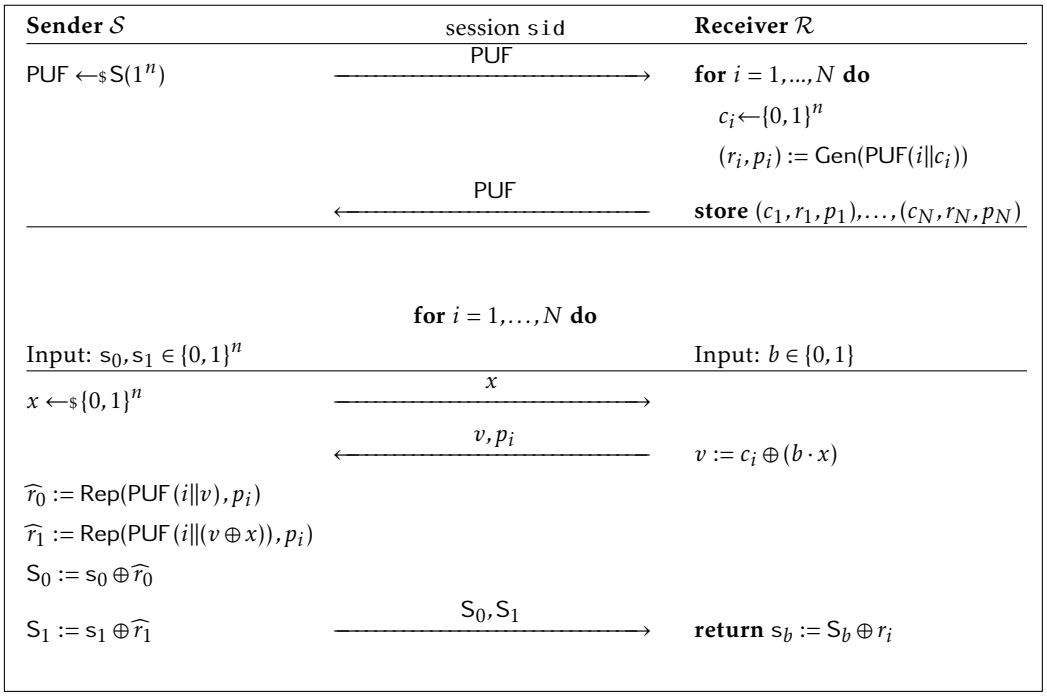


Figure 2.3: Oblivious transfer protocol.

2.4.1 Universally Composable Oblivious Transfer

Our OT protocol adapts the protocol of Brzuska et al. [BFS+11], which is secure against attackers limited to honestly generated PUFs. Essentially, we simply switch which party generates the PUF.

In our description of the protocol in Figure 2.3, we have the parties exchange the PUF once, after which they can subsequently execute any pre-determined number N of oblivious-transfer executions.

Theorem 17. *The protocol depicted in Figure 2.3 securely realizes $\mathcal{F}_{\mathsf{OT}}$ in the $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{auth}})$-hybrid model, where malicious parties are limited to generate stateless PUFs (with arbitrary behavior). The security holds unconditionally.*

The security of our protocol holds in the statistical sense which means that, the environment's view between the ideal and the real wold is statistically close, even if the algorithms $\mathcal{A}$, $\mathcal{Z}$,

$\mathcal{F}_{\mathsf{OT}}$ is parameterized by an integer N and receives as input a security parameter 1^n, and runs with parties $P_1, ..., P_n$ and adversary Sim. The functionality initially sets $(i, S, R) = (1, \perp, \perp)$. In the following, the functionality ignores any input if $i > N$, or if $i > 1$ and $(S, R) \neq (\mathcal{S}, \mathcal{R})$ for the parties' identities $(\mathcal{S}, \mathcal{R})$ in the input. Else,

- Whenever $\mathcal{S}$ sends $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{s}_0, \mathsf{s}_1))$ with $\mathsf{s}_0, \mathsf{s}_1 \in \{0,1\}^n \cup \{\perp\}$ to $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{F}_{\mathsf{OT}}$ stores $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{s}_0, \mathsf{s}_1))$ and sends $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to Sim. The functionality increments i to $i + 1$ and, if now $i = 2$, stores $(S, R) = (\mathcal{S}, \mathcal{R})$.
- Whenever $\mathcal{R}$ sends $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, b)$ for $b \in \{0,1\}$ to $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{F}_{\mathsf{OT}}$ stores this tuple and sends $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{ssid}, \mathtt{sid}, \mathcal{S}, \mathcal{R})$ to Sim.
- When Sim sends $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{F}_{\mathsf{OT}}$ checks if tuples $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{s}_0, \mathsf{s}_1))$ and $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, b)$ have been stored. If so, send $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, \mathsf{s}_b)$ to $\mathcal{R}$.
- When Sim sends $(\mathsf{receipt}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to $\mathcal{F}_{\mathsf{OT}}$, $\mathcal{F}_{\mathsf{OT}}$ checks if the tuple $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, b)$ has been stored. If so, send $(\mathsf{receipt}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to $\mathcal{S}$.

Figure 2.4: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ for oblivious transfer adapted from [Can01].

and Sim are unbounded but query the PUF only a polynomial number of times. The proof follows [BFS+11] closely.

Proof. We prove the security of our construction for static corruptions, which simplifies the proof because we can describe different simulators depending on the corrupted party. We further simplify the exposition of the proof by fixing the number of OTs to $N = 1$, but we stress that the proof straightforwardly carries over to the more general case. In the following we construct an ideal world adversary (simulator) Sim that interacts with the ideal world functionality $\mathcal{F}_{\mathsf{OT}}$ such that no environment $\mathcal{Z}$ can distinguish an interaction with $\mathcal{A}$ in the real world from a protocol execution with Sim in the ideal world. To do so, the simulator Sim invokes a copy of $\mathcal{A}$ and it simulates the interaction of $\mathcal{A}$ with $\mathcal{Z}$ and the parties $\mathcal{S}$ and $\mathcal{R}$.

Destroying PUFs. Malicious parties may destroy PUFs they create in the real world. However, it is clear that this does not help them. Formally, this is because Sim (in the ideal world) can simply ignore such a request (and the adversary has no way of telling whether the "PUF" was destroyed or not. We therefore do not explicitly mention this possibility in what follows.

Simulating the Communication with $\mathcal{Z}$ The simulator Sim forwards the messages between $\mathcal{Z}$ and $\mathcal{A}$. This means that any message that Sim receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape and whenever $\mathcal{A}$ writes on his output tape, then Sim copies this value on its own output tape.

Both parties are honest. The simulator simulates both $\mathcal{S}$ and $\mathcal{R}$ locally and works as follows.

- Whenever $\mathcal{Z}$ writes two secrets $(\mathsf{s}_0, \mathsf{s}_1)$ on the communication tape of $\mathcal{S}$ in the real world, then Sim obtains $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on its input tape. Sim then picks two secrets $(\mathsf{s}'_0, \mathsf{s}'_1)$ uniformly at random and writes them on $\mathcal{S}'s$ input tape. In addition, Sim chooses

two random strings x_0 and x_1 and runs $\mathcal{S}$ which sends $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (x_0, x_1))$ to the simulated functionality $\mathcal{F}_{\mathsf{auth}}$.

- If $\mathcal{F}_{\mathsf{OT}}$ writes $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on Sim's input tape, then Sim chooses a random bit b and writes b on $\mathcal{R}$'s input tape.
- The simulator Sim relays all communications between the simulated parties $\mathcal{S}, \mathcal{R}$, $\mathcal{F}_{\mathsf{PUF}}$, and $\mathcal{A}$ and follows their program as long as they do not produce any local output.
- At some point, the party $\mathcal{R}$ outputs s_b and Sim then writes $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on the input tape of $\mathcal{F}_{\mathsf{OT}}$. If $\mathcal{S}$ outputs $\mathsf{receipt}_{\mathsf{OT}}$, then Sim writes $(\mathsf{receipt}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on $\mathcal{F}_{\mathsf{OT}}$'s input tape.

Recall, that both $\mathcal{Z}$ and $\mathcal{A}$ may have evaluated the PUF in the setup phase a polynomial number of times. In the following we show that tuple $(x, c \oplus (b \cdot x), \mathsf{s}_0 \oplus \widehat{r_0}, \mathsf{s}_1 \oplus \widehat{r_1})$ is uniformly distributed from $\mathcal{A}$'s point of view. To do so, it is sufficient to show that with high probability each variable corresponding to each entry looks random and that each entry does not depend on the previous entry. Obviously, x is a random string, the value $c \oplus (b \cdot x)$ is computed by first choosing c uniformly at random and then computing either c or $c \oplus x$ depending on the value of b. Thus, in both cases this value is distributed independently of x and b. Since the PUFs have a well-spread domain, it follows that the adversary did not query the PUFs on challenges closer than d_{min} from $c \oplus (b \cdot x)$ or $x \oplus c \oplus (b \cdot x)$. If this is the case, then the outputs of $\mathsf{Gen}(\widehat{r_\alpha})$ for $\alpha = 0, 1$ are distributed almost according to U_l. In particular, the statistical distance from U_l is at most $\epsilon(n)$ even after learning the helper data. Thus, their statistical distance is at most $2\epsilon(n)$ which is negligible.

Receiver is corrupted. We turn to the case where $\mathcal{S}$ is honest and $\mathcal{R}$ is malicious. In the setup phase the simulator Sim observes the dishonest receiver queries (asked by $\mathcal{A}$ and $\mathcal{Z}$) to the honestly generated sender PUF and adds all query-answer pairs to a list $\mathcal{L}$. In addition, Sim creates an initially empty list of challenge values $\mathcal{C}$. Note, that the malicious receiver cannot replace the honestly generated PUF with a maliciously generated one, since the sender can simply check the sid of the returned PUF and would reject a replaced PUF.

Whenever $\mathcal{Z}$ writes two secrets $(\mathsf{s}_0, \mathsf{s}_1)$ on the communication tape of $\mathcal{S}$ in the real world, then Sim obtains $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on its input tape. The simulator Sim then follows the protocol description, picks a random value x and sends $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, x)$ to the simulated functionality $\mathcal{F}_{\mathsf{auth}}$. At some point, $\mathcal{A}$ asks the simulated $\mathcal{R}$ to send a value v to $\mathcal{S}$ in the real world, then Sim first checks that $\mathsf{D}(v, \mathcal{C}) > d_{\mathsf{min}}$ and $\mathsf{D}(v \oplus x, \mathcal{C}) > d_{\mathsf{min}}$. If both cases are true, then Sim verifies for each $(c, r) \in \mathcal{L}$ that either the hamming distance of c and v or of c and $v \oplus x$ is smaller than d_{min}. Observe that both cases cannot occur simultaneously because, $\mathsf{D}(c, v) < d_{\mathsf{min}}$ and $\mathsf{D}(c, v \oplus x) < d_{\mathsf{min}}$ implies that $\mathsf{D}(v, v \oplus x) < 2d_{\mathsf{min}}$. However the well-spread domain property guarantees that the latter only happens with negligible probability. Thus, we have proven that only one of the two cases can hold. The next step of the proof is to show that the challenges in $\mathcal{L}$ are either not close to v or are not close to $v \oplus x$. Suppose to the contrary that there exist two queries c_0 and c_1 to the PUF such that $\mathsf{D}(c_0, v) < d_{\mathsf{min}}$ and $\mathsf{D}(c_1, v \oplus x) < d_{\mathsf{min}}$. If this is the case, then $\mathsf{D}(c_0 \oplus x, c_1) < 2d_{\mathsf{min}}$ or equivalently, that $\mathsf{D}(x, c_0 \oplus c_1) \leq 2d_{\mathsf{min}}$. Since $\mathcal{A}$ has made all PUF queries, this means that the value x is a random value that is independent of any value c that $\mathcal{A}$ evaluated the PUF on. This implies that if $\mathcal{A}$ is able to choose c_0, c_1 in such

a way, then this must be true for a non-negligible fraction of values $x \in \{0,1\}^n$. If we denote by $p(n)$ the number of queries by $\mathcal{A}$, then we can compute the number of sums of two different challenges as $\frac{p(n)}{2} = \frac{1}{2}p(n)(p(n)-1)$. This equation represents only a negligible fraction of 2^n. We can further bound this number by multiplying the number of all elements in a ball of radius $2d_{\mathsf{min}}$ to the number above, then we see that this property still holds as $2d_{\mathsf{min}}$ is still in $o(n/\log n)$. This implies that $\mathcal{A}$ can only cover a negligible number of values x.

Now, the simulator sets $b := 0$ if there exists a challenge with $\mathsf{D}(c, v) < d_{\mathsf{min}}$ or $b := 1$ if there exists challenges with $\mathsf{D}(c, v \oplus x) < d_{\mathsf{min}}$. If no such challenges exists then Sim chooses b at random. Afterwards, Sim sends $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{R}, \mathcal{S}, v)$ in the internal simulation and Sim writes $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, v)$ on $\mathcal{F}_{\mathsf{OT}}$'s input tape. When Sim gets the message $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$, then Sim writes $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on $\mathcal{F}_{\mathsf{OT}}$'s input tape obtaining $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, s_b)$ on the malicious $\mathcal{R}$'s input tape, which is simulated by Sim for $\mathcal{A}$. To compute the answer for the message $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{R}, \mathcal{S}, v)$, Sim chooses a random secret s_{1-b}, and the PUF and Gen to compute $(\widehat{r_\alpha}, p_\alpha)$, for $\alpha = 0, 1$ following the protocol. It then sends $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{s}_0 \oplus \widehat{r_0}, p_0, \mathsf{s}_1 \oplus \widehat{r_1}, p_1))$ to the simulated $\mathcal{F}_{\mathsf{auth}}$.

The last step of this part of the proof is to show that the joint views of $\mathcal{Z}$ and $\mathcal{A}$ in the protocol execution in the real world is indistinguishable from the joint view of $\mathcal{Z}$ and $\mathcal{A}$ (simulated by Sim) in the ideal world. Obviously, both joint views differ only in the last message received by $\mathcal{R}$. Since the adversary never queried the PUF on any challenge with distance smaller than d_{min} from $v \oplus ((1-b) \cdot x)$, it follows by the response independence of the honest PUF that $\widehat{r_{1-b}}$ is quasi uniform even if p_{1-b} is known. Thus, from $\mathcal{A}$'s point of view the last messages are distributed identically.

Sender is corrupted. We consider the case where $\mathcal{S}$ is malicious and $\mathcal{R}$ is honest. In this case we build a simulator Sim that extracts both secrets $\mathsf{s}_0, \mathsf{s}_1$ from the malicious sender using its permanent access to the PUFs. In the setup phase Sim follows the protocol description of the honest $\mathcal{R}$ and to extracts both secrets, the simulator waits that the following two events happen:

1. $\mathcal{A}$ asks $\mathcal{S}$ to send $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, x)$ in the simulation.

2. The ideal functionality $\mathcal{F}_{\mathsf{OT}}$ sends $(\mathsf{choose-secret}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to Sim.

If both events happen in arbitrary order, then Sim chooses a random string $v \leftarrow \{0,1\}^n$, sends $(\mathsf{send}_{\mathsf{auth}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, v)$ to $\mathcal{S}$, and writes $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ to the input tape of $\mathcal{F}_{\mathsf{OT}}$. The simulator then evaluates the PUF computing $\overline{r}_\alpha$ for $\alpha \in \{0,1\}$ by sending $(\mathsf{eval}_{\mathsf{PUF}}, \mathsf{PUF}, \mathcal{S}, v \oplus (\alpha \cdot x))$ to the ideal functionality $\mathcal{F}_{\mathsf{PUF}}$. Afterwards, Sim uses the helper data p_α and the algorithm Rep to obtain $\widehat{r_\alpha}$. If $\widehat{r_\alpha} := \mathsf{Rep}(\overline{r}_\alpha, p_\alpha)$ fails for $\alpha \in \{0,1\}$, then set s_α to $\perp$. If all executions succeeded, then Sim sets $\mathsf{s}_0 := \widehat{r_0} \oplus a_0$ and $\mathsf{s}_1 := \widehat{r_1} \oplus a_1$. Afterwards, Sim writes $(\mathsf{send}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{s}_0, \mathsf{s}_1))$ on $\mathcal{F}_{\mathsf{OT}}$'s input tape and Sim also writes $(\mathsf{deliver}_{\mathsf{OT}}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S}, \mathcal{R})$ on $\mathcal{F}_{\mathsf{OT}}$'s input tape.

Finally, we have to show that the simulation of the malicious $\mathcal{S}$ in the ideal world is indistinguishable from the real world execution. This is true because v is random string in both worlds and also because the output of $\mathcal{R}$ is the same in both worlds. To see this, observe that Sim computes the output as $\mathcal{R}$ does it in the real world for both possible choice bits. □

2.4.2 From UC Oblivious Transfer to UC Two-Party Computation

We observe that our UC oblivious-transfer protocol can be used to obtain UC two-party computation of any functionality. The main idea is to first construct a semi-honest secure two-party computation protocol using Yao's garbled-circuit protocol, and to then apply the compiler of Ishai, Prabhakaran, and Sahai [IPS08].

Semi-honest secure two-party computation. Lindell and Pinkas presented a proof for Yao's two-party secure-computation protocol [LP09]. They show how to instantiate the garbling part of the protocol with a private-key encryption scheme having certain special properties. In addition, the authors show that any pseudorandom function is sufficient to instantiate such a private-key encryption scheme. Our main observation is that we can replace the pseudorandom function with a PUF.[4] This has already been observed before by Brzuska et al. [BFS+11] in a different context. With this observation, we can apply the result of [LP09] to obtain a protocol for semi-honest secure two-party computation based on PUFs only (and no computational assumptions).

Theorem 18. *Let f be any functionality. Then there is a (constant-round) protocol that securely computes f for semi-honest adversaries in the $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{OT}})$-hybrid model.*

We omit the proof since it follows easily from prior work.

Universally composable two-party computation. In the next step we apply the IPS compiler [IPS08], a black-box compiler that takes

1. An "outer" MPC protocol Π with security against a constant fraction of malicious parties.
2. An "inner" two-party protocol ρ, in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, where the security of ρ only needs to hold against semi-honest parties.

and transforms them into a two-party protocol $\Phi_{\Pi,\rho}$ which is secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model against malicious corruptions.

In our setting, we must be careful to give information-theoretic instantiations of the "outer" and "inner" protocols so that our final protocol $\Phi_{\Pi,\rho}$ will be unconditionally secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model. Fortunately, we may instantiate the "outer" protocol, Π, with the seminal BGW protocol [BGW88] and may instantiate the "inner" protocol, ρ, with the protocol from the previous section. Alternatively, the "inner" protocol can be instantiated with the semi-honest version of the two-party GMW protocol [GMW87] in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model.

Let ψ denote the OT-protocol described in Figure 2.3 and let $\Phi^{\psi}_{\Pi,\rho}(f)$ denote the IPS-compiled protocol which makes subroutine calls to ψ instead of $\mathcal{F}_{\mathsf{OT}}$ and computes the functionality f. Using Theorem 17 and Theorem 18, along with the UC composition theorem, we obtain the following result:

Theorem 19. *For any functionality f, protocol $\Phi^{\psi}_{\Pi,\rho}(f)$ securely computes f in the $(\mathcal{F}_{\mathsf{PUF}}, \mathcal{F}_{\mathsf{auth}})$-hybrid model.*

[4] Note also that if the circuit generator is malicious, then he cannot violate the circuit evaluator's privacy by generating a malicious PUF.

3 On Tight Security Proofs for Schnorr Signatures

3.1 Motivation

The security of a cryptosystem is nowadays usually confirmed by giving a security proof. Typically, such a proof describes a *reduction* from some (assumed-to-be-)hard computational problem to breaking a defined security property of the cryptosystem. A reduction is considered as *tight*, if the reduction solving the hard computational problem has essentially the same running time and success probability as the attacker on the cryptosystem. Essentially, a tight reduction means that a successful attacker can be turned into an efficient algorithm for the hard computational problem *without* any significant increase in the running time and/or significant loss in the success probability. The tightness of a reduction thus determines the strength of the security guarantees provided by the security proof: A non-tight reduction gives weaker security guarantees than a tight one. Moreover, tightness of the reduction affects the efficiency of the cryptosystem when instantiated in practice: A tighter reduction allows to securely use smaller parameters (shorter moduli, a smaller group size, etc.). Therefore, it is very desirable for a cryptosystem to have a tight security reduction. What is considered tight or non-tight may differ depending on the circumstances, but usually even a polynomially-bounded increase/loss is considered as significant, if the polynomial may be large. An increase/loss by a small constant factor on the other hand is not considered as significant.

In the domain of digital signatures, tight reductions are known for many fundamental schemes, such as Rabin/Williams signatures [Ber08], many strong-RSA-based signatures [Sch11], and RSA Full-Domain Hash [KK12]. For Schnorr signatures [Sch90; Sch91], however, the story is a bit different. Schnorr's scheme is one of the most fundamental public-key cryptosystems and Pointcheval and Stern have shown that it is provably secure, assuming the hardness of the discrete logarithm (DL) problem [PS96] in the Random Oracle Model (ROM) [BR93]. However, the reduction of Pointcheval and Stern from the discrete logarithm problem to breaking Schnorr signatures is not tight: It loses a factor of q in the time-to-success ratio, where q is the number of random oracle queries performed by the forger.

This has lead to a long line of research investigating the existence of tighter security proofs for Schnorr signatures. At Asiacrypt 2005 Paillier and Vergnaud [PV05] gave a first lower bound showing that any algebraic reduction (even in the ROM) converting a forger for Schnorr signatures into an algorithm solving the discrete logarithm problem must lose a factor of at least $q^{1/2}$. Their result is quite strong, as they rule out reductions even for adversaries that do not have access to a signing oracle and receive as input the message for which they must forge (UUF-NMA, see Section 3.3.1.1 for a formal definition). However, their result also has some limitations: It holds only under the interactive one-more discrete logarithm (OMDL) assumption, they only consider algebraic reductions, and they only rule out tight reductions from the (one-

more) discrete logarithm problem. At Crypto 2008 Garg et al. [GBL08] refined this result, by improving the bound from $q^{1/2}$ to $q^{2/3}$ with a new analysis and show that this bound is optimal if the meta-reduction follows a particular approach for simulating the forger. At Eurocrypt 2012 Seurin [Seu12] finally closed the gap between the security proof of Pointcheval and Stern [PS96] and known impossibility results, by describing a novel elaborate simulation strategy for the forger and providing a new analysis. Thus, to summarize, all previous works [PV05; GBL08; Seu12] on the existence of tight security proofs for Schnorr signatures have the following in common:

1. They only rule out the existence of tight reductions from specific strong computational problems, namely the (one-more) discrete logarithm problem [BNP+03]. Reduction from weaker problems such as, e.g., the computational or decisional Diffie-Hellman problem (CDH/DDH) are not considered.

2. The impossibility results are not unconditional but instead are themselves only valid under the very strong OMDL hardness assumption.

3. They hold only with respect to a limited (but natural) class of reductions, so-called *algebraic reductions*.

It is not entirely unlikely that first the nonexistence of a tight reduction from *strong* computational problems is proven, and later a tight reduction from some *weaker* problem is found. A concrete recent example in the domain of digital signatures where this has happened is RSA Full-Domain Hash (RSA-FDH) [BR96]. First, at Crypto 2000 Coron [Cor00] described a non-tight reduction from solving the RSA-problem to breaking the security of RSA-FDH, and at Eurocrypt 2002 [Cor02a] showed that under certain conditions no tighter reduction from RSA can exist. Later, at Eurocrypt 2012, Kakvi and Kiltz [KK12] gave a tight reduction from solving a weaker problem, the so-called Phi-Hiding problem. The leverage, used by Kakvi and Kiltz to circumvent the aforementioned impossibility results, was to assume hardness of a weaker computational problem, i.e., making a stronger assumption. As all previous works rule out only tight reductions from strong computational problems such as DL and OMDL, this might happen again with Schnorr signatures and the question whether a tight security reduction from weaker problems might exist was left open for 25 years. A very first non-trivial *tight* security reduction for Schnorr signatures was recently presented by Kiltz, Masny, and Pan [KMP16]. However, the necessary assumption is interactive, highly non-standard, and is specifically tailored to allow the proof to go through. The following question therefore still remained open:

> *Does a tight security proof for Schnorr signatures exist based on any weaker non-interactive computational problem?*

3.2 Contribution

In this chapter we answer this question in the negative for an overwhelming class of weaker problems, ruling out the existence of tight reductions for virtually all natural non-interactive computational problems defined over abstract algebraic groups. Like previous works, we consider universal unforgeability under no-message attacks (UUF-NMA-security). Moreover, our results hold unconditionally. In contrast to previous works, we consider *generic* reductions

instead of algebraic reductions, but we believe that this restriction is marginal: The motivation of considering only algebraic reductions from [PV05] applies equally to generic reductions. In particular, to the best of our knowledge all known examples of algebraic reductions are also generic.

Our main technical contribution is a new approach for the simulation of a forger in a meta-reduction, i.e., "a reduction against the reduction", which differs from previous works [PV05; GBL08; Seu12] and which allows us to show the following main result:

Theorem (informal)**.** *For almost any* natural *non-interactive computational problem* Π*, there is no tight* generic *reduction from solving* Π *to breaking the universal unforgeability under no-message attacks of Schnorr signatures.*

Technical Approach. We begin with the hypothesis that there exists a tight generic reduction $\mathcal{R}$ from some hard non-interactive problem Π to the universal unforgeability under no-message attacks (UUF-NMA) of Schnorr signatures. Then we show that under this hypothesis there exists an efficient algorithm $\mathcal{M}$, a meta-reduction, which efficiently solves Π. This implies that the hypothesis is false. The meta-reduction $\mathcal{M} = \mathcal{M}^{\mathcal{R}}$ runs $\mathcal{R}$ as a subroutine, by efficiently *simulating* the forger $\mathcal{A}$ for the reduction $\mathcal{R}$.

The difficulty with meta-reductions is that $\mathcal{M} = \mathcal{M}^{\mathcal{R}}$ must efficiently *simulate* the forger $\mathcal{A}$ for $\mathcal{R}$. All previous works in this direction [PV05; GBL08; Seu12] followed essentially the same approach, using a discrete logarithm oracle provided by the OMDL assumption. This oracle allows to efficiently compute valid signatures in the simulation of forger $\mathcal{A}$. This is also the reason why all previous results are only valid under the OMDL assumption, and were only able to rule out reductions from the discrete log or the OMDL problem. To overcome these limitations, a new simulation technique is necessary.

We revisit the simulation strategy of $\mathcal{A}$ applied in known meta-reductions, and put forward a new technique for proving impossibility results. It turns out that considering *generic* reductions provides additional leverage for simulating a successful forger efficiently, essentially by suitably re-programming the group representation while computing valid signatures. The technical challenge is to prove that the reduction remains oblivious to theses changes to the group representation during the simulation, except for some negligible probability. We show how to prove this by adopting the "low polynomial degree" proof technique of Shoup [Sho97], which was originally introduced to analyze the complexity of certain algorithms for the discrete logarithm problem, to the setting considered in this paper.

This new approach turns out to be extremely powerful, as it allows to rule out reductions from any non-interactive *representation-invariant* computational problem. Since almost all common hardness assumptions in algebraic groups (e.g., DL, CDH, DDH, DLIN, etc.) are based on representation-invariant computational problems, we are able to rule out tight generic reductions from virtually any natural computational problem, without making any additional assumptions. Even though we apply it specifically to Schnorr signatures, the overall approach is general. We expect that it is applicable to other cryptosystems as well.

Generic Reductions vs. Algebraic Reductions Similar to algebraic reductions, a generic reduction performs only group operations. The main difference is that the sequence of group operations performed by an algebraic reduction may (but, to our best knowledge, in all known

examples does not) depend on the particular representation of group elements. A generic reduction, however, is required to work essentially identical for any representation of group elements. Generic reductions are by definition more restrictive than algebraic ones.

An obvious question arising with our work is the relation between algebraic and generic reductions. Is a lower bound for generic reductions much less meaningful than a bound for algebraic reductions? We argue that the difference is not very significant. The restriction to algebraic reductions was motivated by the fact that most reductions in known security proofs treat the group as a black-box, and thus are algebraic [PV05; GBL08; Seu12]. However, the same motivation applies to generic reductions as well, with exactly the same arguments. In particular, virtually all examples of algebraic reductions in the literature are also generic.

The vast majority of reductions in common security proofs for group-based cryptosystems treats the underlying group as a black-box (i.e., works for any representation of the group), and thus is generic. This is a very desirable feature, because then the cryptosystem can securely be instantiated with *any* group in which the underlying computational problem is hard. In contrast, representation-specific security proofs would require to re-prove security for any particular group representation the scheme is used with. Therefore considering generic reductions seems very reasonable.

Generic Reductions vs. Security Proofs in the Generic Group Model. We stress that we model only the reduction $\mathcal{R}$ as a generic algorithm. We do not restrict the forger $\mathcal{A}$ in this way, as commonly done in security proofs in the generic group model. It may not be obvious that this is possible, because $\mathcal{A}$ expects as input group elements in some specific encoding, while $\mathcal{R}$ can only specify them in the form of random encodings. However, the reduction only gets access to the adversary as a black-box, which means that the adversary is external to the reduction, and the environment in which the reduction runs can easily translate between the encodings used by the reduction and the adversary. Further, note that while some reduction from a problem Π may be generic, the actual algorithm solving said problem is not $\mathcal{R}$ itself, but the composition of $\mathcal{R}$ and $\mathcal{A}$ which may very well be non-generic. In particular, this means that any results about equivalence of interesting problems in the generic group model do not apply to the reduction. See Section 3.3.4 and Figure 3.2 for further explanation.

Generic Reductions in the Non-Programmable Random Oracle Model. An orthogonal question to the one answered in our main result is whether security proofs – even non-tight ones – for Schnorr signatures exist in weaker models. In another result of [PV05] Paillier and Vergnaud analyzed the security of Schnorr Signatures in the standard model. In particular, they presented an impossibility result for security proofs based on algebraic reductions and the discrete logarithm problem. In a similar vein, Fischlin and Fleischhacker [FF13b] presented a result about the security of Schnorr signatures in the non-programmable random oracle model. Essentially they prove that in the *non-programmable* ROM [FLR+10] no reduction from the discrete logarithm problem exists that invokes the adversary only ever on the same input. This class is limited, but encompasses all forking-lemma style reductions used to prove Schnorr signatures secure in the programmable ROM.

Both results suffer from the same shortcomings already discussed earlier. They only show impossibility for the discrete logarithm problem and they are themselves not unconditional, in that they rely on the hardness of the one-more discrete logarithm problem. By applying our new

simulation technique to reductions in the non-programmable random oracle model, we continue this line of research and show the following result:

Theorem (informal)**.** *For almost any* natural *non-interactive computational problem* Π*, there is no (even non-tight)* generic *reduction from solving* Π *to breaking the universal unforgeability under no-message attacks of Schnorr signatures in the non-programmable random oracle model.*

Further Related Work. Dodis et al. [DHT12] showed that it is impossible to reduce any computational problem to breaking the security of RSA-FDH in a model where the RSA-group $\mathbb{Z}_N^*$ is modeled as a generic group. This result extends [DOP05]. Coron [Cor02a] considered the existence of tight security reductions for RSA-FDH signatures [BR96]. This result was generalized by Dodis and Reyzin [DR03] and later refined by Kiltz and Kakvi [KK12].

In the context of Schnorr signatures, Neven et al. [NSW09] described necessary conditions the hash function must meet in order to provide existential unforgeability under chosen-message attacks (EUF-CMA), and showed that these conditions are sufficient if the forger (not the reduction!) is modeled as a generic group algorithm.

3.3 Preliminaries

In this section we describe Schnorr's signature scheme and introduce general definitions for security for signature schemes, computational problems and generic reductions.

3.3.1 Digital Signatures and Unforgeability

We first recall the syntax and correctness condition of a digital signature scheme, and the definition of Schnorr's signature scheme.

Definition 20. *A signature scheme* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *consists of three algorithms:*

The key generation algorithm KGen *takes as input the security parameter* 1^n *and generates a key pair* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(1^n)$.

The probabilistic signing algorithm Sig *takes as input a secret key* sk *and a message* $m \in \{0,1\}^*$ *and outputs a signature* $\sigma \leftarrow_\$ \mathsf{Sig}(\mathsf{sk}, m)$.

The deterministic verification algorithm Vf *takes as input a public key* pk*, a message* m*, and a candidate signature* σ *and outputs a bit* $b \leftarrow \mathsf{Vf}(\mathsf{pk}, m, \sigma)$.

The scheme is correct *if and only if for all* $n \in \mathbb{N}$*, all* $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(1^n)$*, all* $m \in \{0,1\}^*$*, and all* $\sigma \leftarrow_\$ \mathsf{Sig}(\mathsf{sk}, m)$*, it holds that* $\mathsf{Vf}(\mathsf{pk}, m, \sigma) \stackrel{?}{=} 1$.

Definition 21. *Let* $\mathbb{G}$ *be a group of order* p *with generator* g*, and let* $\mathsf{H} : \mathbb{G} \times \{0,1\}^\ell \to \mathbb{Z}_p$ *be a hash function. The Schnorr signature scheme [Sch90; Sch91] consists of the following efficient algorithms* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$.

$\mathsf{KGen}(g)$**:** *The key generation algorithm takes as input a generator* g *of* $\mathbb{G}$*. It chooses* $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p$*, computes* $\mathsf{pk} := g^{\mathsf{sk}}$*, and outputs* $(\mathsf{pk}, \mathsf{sk})$.

$\mathsf{Sig}(\mathsf{sk}, m)$**:** *The input of the signing algorithm is a private key* sk *and a message* $m \in \{0,1\}^\ell$*. It chooses a random integer* $r \leftarrow_\$ \mathbb{Z}_p$*, sets* $R := g^r$ *as well as* $c := \mathsf{H}(R, m)$*, and computes* $y := \mathsf{sk} \cdot c + r \bmod p$*. It outputs* $\sigma = (R, y)$.

$\mathsf{Vf}(\mathsf{pk}, m, (R, y))$**:** *The verification algorithm outputs the truth value of* $g^y \stackrel{?}{=} \mathsf{pk}^c \cdot R$*, where* $c = \mathsf{H}(R, m)$*.*

3.3.1.1 Unforgeability

We next define universal unforgeability under no-message attacks. Consider the following security experiment involving a signature scheme $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$, an attacker $\mathcal{A}$, and a challenger $\mathcal{C}$.

1. The challenger $\mathcal{C}$ computes a key-pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KGen}(g)$ and chooses a message $m \leftarrow_\$ \{0,1\}^\ell$ uniformly at random. It invokes $\mathcal{A}$ on input (pk, m).

2. Eventually, $\mathcal{A}$ stops, outputting a signature σ.

Definition 22. *We say that* $\mathcal{A}$ (ϵ, t)*-breaks the* UUF-NMA*-security of* $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$*, if* $\mathcal{A}$ *runs in time at most* t *and*

$$\Pr\left[\mathcal{A}(\mathsf{pk}, m) = \sigma : \mathsf{Vf}(\mathsf{pk}, m, \sigma) = 1\right] \geq \epsilon,$$

where randomness is taken over the random choice of pk*,* m*, and* $\mathcal{A}$*'s random coins.*

Note that UUF-NMA-security is a very weak security goal for digital signatures. Since we are going to prove a negative result, this is not a limitation, but instead, makes our result even stronger. In fact, if we rule out reductions from some problem Π to forging signatures in the sense of UUF-NMA, then the impossibility clearly holds for stronger security notions, such as existential unforgeability under adaptive chosen-message attacks [GMR88], too.

3.3.2 Computational Problems

Let $\mathbb{G}$ be a cyclic group of order p and $g \in \mathbb{G}$ a generator of $\mathbb{G}$. We write $\mathsf{desc}(\mathbb{G}, g)$ to denote the list of group elements $\mathsf{desc}(\mathbb{G}, g) = (g, g^2, \ldots, g^p) \in \mathbb{G}^p$. We say that $\mathsf{desc}(\mathbb{G}, g)$ is the *enumerating description* of $\mathbb{G}$ with respect to g.

Definition 23. *A non-interactive* computational problem Π *in* $\mathbb{G}$ *is specified by two (computationally unbounded) procedures* $\Pi = (\mathcal{G}_\Pi, \mathcal{V}_\Pi)$*, with the following syntax.*

$\mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$ *takes as input an enumerating description of* $\mathbb{G}$*, and outputs a state* st *and a problem instance (the challenge)* $C = (C_1, \ldots, C_u, C') \in \mathbb{G}^u \times \{0,1\}^*$*. We assume in the sequel that at least* C_1 *is a generator of* $\mathbb{G}$*.*

$\mathcal{V}_\Pi(\mathsf{desc}(\mathbb{G}, g), st, S, C)$ *takes as input* $(\mathsf{desc}(\mathbb{G}, g), st, C)$ *as defined above, and* $S = (S_1, \ldots, S_w, S') \in \mathbb{G}^w \times \{0,1\}^*$*. It outputs* 0 *or* 1*.*

The exact description and distribution of st, C, S *depends on the considered computational problem.*

Definition 24. *An algorithm* $\mathcal{A}$ (ϵ, t)*-solves the non-interactive computational problem* Π *if* $\mathcal{A}$ *has running time at most* t *and wins the following interactive game against a (computationally unbounded) challenger* $\mathcal{C}$ *with probability at most* ϵ*, where the game is defined as follows:*

1. *The challenger* $\mathcal{C}$ *generates an instance of the problem* $(st, C) \leftarrow_\$ \mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$ *and invokes* $\mathcal{A}$ *on input* C*.*

2. *Eventually, algorithm $\mathcal{A}$ outputs a candidate solution S. The algorithm $\mathcal{A}$ wins the game (i.e., solves the computational problem correctly) if and only if $\mathcal{V}_\Pi(\mathsf{desc}(\mathbb{G}, g), st, C, S) = 1$.*

Example 25. The *discrete logarithm problem* in $\mathbb{G}$ is specified by the following procedures. The sampling algorithm $\mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$ outputs (st, C) with $st = \emptyset$ and $C = (g, h)$, where $h \leftarrow_\$ \mathbb{G}$ is a random group element. The verification algorithm $\mathcal{V}_\Pi(\mathsf{desc}(\mathbb{G}, g), st, C, S)$ interprets $S = S' \in \{0,1\}^*$ canonically as an integer in $\mathbb{Z}_p$, and outputs 1 iff $h = g^{S'}$.

Example 26. The UUF-NMA-forgery problem for Schnorr signatures in $\mathbb{G}$ with hash function H is specified by the following procedures. $\mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$ outputs (st, C) with $st = m$ and $C = (g, \mathsf{pk}, m) \in \mathbb{G}^2 \times \{0,1\}^\ell$, where $\mathsf{pk} = g^{\mathsf{sk}}$ for $\mathsf{sk} \leftarrow_\$ \mathbb{Z}_p$ and $m \leftarrow_\$ \{0,1\}^\ell$. The verification algorithm $\mathcal{V}_\Pi(\mathsf{desc}(\mathbb{G}, g), st, C, S)$ parses S as $S = (R, y) \in \mathbb{G} \times \mathbb{Z}_p$, sets $c := \mathsf{H}(R, st)$, and outputs 1 if and only if $\mathsf{pk}^c \cdot R = g^y$.

3.3.3 Representation-Invariant Computational Problems

In our impossibility results given below, we want to rule out the existence of a tight reduction from as large a class of computational problems as possible. Ideally, we want to rule out the existence of a tight reduction from any computational problem that meets Definition 23. However, it is easy to see that this is not achievable in this generality: as Example 26 shows, the problem of forging Schnorr signatures itself is a problem that meets Definition 23. However, of course there exists a trivial tight reduction from the problem of forging Schnorr signatures to the problem of forging Schnorr signatures! Therefore we need to restrict the class of considered computational problems to exclude such trivial, artificial problems.

We introduce the notion of *representation-invariant* computational problems. Intuitively, a computational problem is *representation-invariant*, if a valid solution to a given problem instance remains valid even if the representation of group elements in challenges and solutions is converted to a different representation of the same group.

This class of problems captures most computational problems defined over an abstract algebraic group. In particular, the problem of forging Schnorr signatures is *not* contained in this class (see Example 29 below).

More formally we define representation-invariance as follows:

Definition 27. *Let $\mathbb{G}, \hat{\mathbb{G}}$ be groups such that there exists an isomorphism $\phi : \mathbb{G} \to \hat{\mathbb{G}}$. We say that Π is* representation-invariant, *if and only if for all isomorphic groups $\mathbb{G}, \hat{\mathbb{G}}$ and for all generators $g \in \mathbb{G}$, all $C = (C_1, \ldots, C_u, C') \leftarrow_\$ \mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$, all $st = (st_1, \ldots, st_t, st') \in \mathbb{G}^t \times \{0,1\}^*$, and all $S = (S_1, \ldots, S_w, S') \in \mathbb{G}^w \times \{0,1\}^*$ holds that $\mathcal{V}_\Pi(\mathsf{desc}(\mathbb{G}, g), st, C, S) = 1 \iff \mathcal{V}_\Pi(\mathsf{desc}(\hat{\mathbb{G}}, \hat{g}), \hat{st}, \hat{C}, \hat{S}) = 1$, where $\hat{g} = \phi(g) \in \mathbb{G}'$, $\hat{C} = (\phi(C_1), \ldots, \phi(C_u), C')$, $\hat{st} = (\phi(st_1), \ldots, \phi(st_t), st')$, and $\hat{S} = (\phi(S_1), \ldots, \phi(S_w), S')$.*

Observe that this definition only demands the *existence* of an isomorphism $\phi : \mathbb{G} \to \hat{\mathbb{G}}$ and not that it is efficiently computable.

Example 28. The discrete logarithm problem is representation-invariant. Let $C = (g, h) \in \mathbb{G}^2$ be a discrete log challenge, with corresponding solution $S' \in \{0,1\}^*$ such that S' canonically interpreted as an integer $S' \in \mathbb{Z}_p$ satisfies $g^{S'} = h \in \mathbb{G}$. Let $\phi : \mathbb{G} \to \hat{\mathbb{G}}$ be an isomorphism, and let $(\hat{g}, \hat{h}) := (\phi(g), \phi(h))$. Then it clearly holds that $\hat{g}^{\hat{S}'} = \hat{h}$, where $\hat{S}' = S'$.

$\mathcal{O}(e, e', \circ)$	$\textsc{GetIdx}(\vec{e})$	$\textsc{Encode}(G)$
$(e, e', \circ) \in E \times E \times \{\cdot, \div\}$	**parse** $\vec{e}$ **as** $(e_1, \ldots, e_w)$	**parse** G **as** $(G_1, \ldots, G_u)$
$(i, j) := \textsc{GetIdx}(e, e')$	**for** $j = 1, \ldots, w$ **do**	**for** $j = 1, \ldots, u$ **do**
return $\textsc{Encode}(\mathcal{L}_i^{\mathbb{G}} \circ \mathcal{L}_j^{\mathbb{G}})$	pick first $i \in \left[\left\lvert\mathcal{L}^E\right\rvert\right]$ s.t. $\mathcal{L}_i^E = e_j$	**if** $\exists i$ s.t. $\mathcal{L}_i^{\mathbb{G}} = G_j$
	$i_j := i$	$e_j := \mathcal{L}_i^E$
	return $(i_1, \ldots, i_w)$	**else**
		$e_j \leftarrow\!\!\$\, E \setminus \mathcal{L}^E$
		append e_j **to** $\mathcal{L}^E$
		append G_j **to** $\mathcal{L}^{\mathbb{G}}$
		return $(e_1, \ldots, e_u)$

Figure 3.1: Procedures implementing the generic group oracle.

Virtually all common hardness assumptions in algebraic groups are based on representation-invariant computational problems. Popular examples are, for instance, the discrete log problem (DL), computational Diffie-Hellman (CDH), decisional Diffie-Hellman (DDH), decision linear (DLIN), and so on.

Example 29. The UUF-NMA-forgery problem for Schnorr signatures with hash function H is *not* representation-invariant for any hash function H. Let $C = (g, \mathsf{pk}, m) \leftarrow\!\!\$\, \mathcal{G}_\Pi(\mathsf{desc}(\mathbb{G}, g))$ be a challenge with solution $S = (R, y) \in \mathbb{G} \times \mathbb{Z}_p$ satisfying $\mathsf{pk}^c \cdot R = g^y$, where $c := \mathsf{H}(R, m)$.

Let $\hat{\mathbb{G}}$ be a group isomorphic to $\mathbb{G}$, such that $\mathbb{G} \cap \hat{\mathbb{G}} = \emptyset$ (that is, there exists no element of $\hat{\mathbb{G}}$ having the same representation as some element of $\mathbb{G}$).[1] Let $\mathbb{G} \to \hat{\mathbb{G}}$ denote the isomorphism. If there exists any R such that $\mathsf{H}(R, m) \neq \mathsf{H}(\phi(R), m)$ in $\mathbb{Z}_p$ (which holds in particular if H is collision resistant and ϕ efficiently computable), then we have

$$g^y = \mathsf{pk}^{\mathsf{H}(R,m)} \cdot R \qquad \text{but} \qquad \phi(g)^y \neq \phi(\mathsf{pk})^{\mathsf{H}(\phi(R),m)} \cdot \phi(R).$$

Thus, a solution to this problem is valid only with respect to a particular given representation of group elements.

The UUF-NMA-forgery problem of Schnorr signatures is not representation-invariant, because a solution to this problem involves the hash value $\mathsf{H}(R, m)$ that depends on a concrete representation of group element R. We consider such complexity assumptions as rather unnatural, as they are usually very specific to certain constructions of cryptosystems.

3.3.4 Generic Reductions

In this section we recall the notion of *generic groups*, loosely following [Sho97] (cf. also [Mau05; RLB+08], for instance), and define generic (i.e., representation independent) reductions.

[1] Such a group $\hat{\mathbb{G}}$ can trivially be obtained for any group $\mathbb{G}$, for instance by modifying the encoding by prepending a suitable fixed string to each group element, and changing the group law accordingly.

Generic Groups. Let $(\mathbb{G},\cdot)$ be a group of order p and $E \subseteq \{0,1\}^{\lceil \log p \rceil}$ be a set of size $|E| = |\mathbb{G}|$. If $g,h \in \mathbb{G}$ are two group elements, then we write $g \div h$ for $g \cdot h^{-1}$. Following [Sho97] we define an *encoding function* as a random injective map $\phi : \mathbb{G} \to E$. We say that an element $e \in E$ is the *encoding* assigned to group element $h \in \mathbb{G}$, if $\phi(h) = e$.

A *generic group algorithm* is an algorithm $\mathcal{R}$ which takes as input $\hat{C} = (\phi(C_1),\ldots,\phi(C_u),C')$, where $\phi(C_i) \in E$ is an encoding of group element C_i for all $i \in [u]$, and $C' \in \{0,1\}^*$ is a bit string. The algorithm outputs $\hat{S} = (\phi(S_1),\ldots,\phi(S_w),S')$, where $\phi(S_i) \in E$ is an encoding of group element S_i for all $i \in [w]$, and $S' \in \{0,1\}^*$ is a bit string. In order to perform computations on encoded group elements, algorithm $\mathcal{R} = \mathcal{R}^{\mathcal{O}}$ may query a *generic group oracle* (or *"group oracle"* for short). This oracle $\mathcal{O}$ takes as input two encodings $e = \phi(G), e' = \phi(G')$ and a symbol $\circ \in \{\cdot,\div\}$, and returns $\phi(G \circ G')$. Note that $(E,\cdot_{\mathcal{O}})$, where $\cdot_{\mathcal{O}}$ denotes the group operation on E induced by oracle $\mathcal{O}$, forms a group which is isomorphic to $(\mathbb{G},\cdot)$.

It will later be helpful to have a specific implementation of $\mathcal{O}$. We will therefore assume in the sequel that $\mathcal{O}$ internally maintains two lists $\mathcal{L}^{\mathbb{G}} \subseteq \mathbb{G}$ and $\mathcal{L}^{E} \subseteq E$. These lists define the encoding function ϕ as $\mathcal{L}^{E}_i = \phi(\mathcal{L}^{\mathbb{G}}_i)$, where $\mathcal{L}^{\mathbb{G}}_i$ and $\mathcal{L}^{E}_i$ denote the i-th element of $\mathcal{L}^{\mathbb{G}}$ and $\mathcal{L}^{E}$, respectively, for all $i \in \left[\left|\mathcal{L}^{\mathbb{G}}\right|\right]$. Note that from the perspective of a generic group algorithm it makes no difference whether the encoding function is fixed at the beginning or lazily evaluated whenever a new group element occurs. We will assume that the oracle uses lazy evaluation to simplify our discussion and avoid unnecessary steps for achieving polynomial runtime of our meta-reductions. The specific implementation of the oracle is shown in Figure 3.1 and explained in detail in the following:

Procedure Encode takes a list $G = (G_1,\ldots,G_u)$ of group elements as input. It checks for each $G_j \in L$ if an encoding has already been assigned to G_j, i.e., whether there exists an index i such that $\mathcal{L}^{\mathbb{G}}_i = G_j$. If this holds, Encode sets $e_j := \mathcal{L}^{E}_i$. Otherwise (if no encoding has been assigned to G_j so far), it chooses a fresh and random encoding $e_j \leftarrow_{\$} E \setminus \mathcal{L}^{E}$. In either case G_j and e_j are appended to $\mathcal{L}^{\mathbb{G}}$ and $\mathcal{L}^{E}$, respectively, which gradually defines the map ϕ such that $\phi(G_j) = e_j$. Note also that the same group element and encoding may occur multiple times in the list. Finally, the procedure returns the list $(e_1,\ldots,e_u)$ of encodings.

Procedure GetIdx takes a list $(e_1,\ldots,e_w)$ of encodings as input. For each $j \in [w]$ it defines i_j as the smallest[2] index such that $e_j = \mathcal{L}^{E}_{i_j}$, and returns $(i_1,\ldots,i_w)$.[3]

The lists $\mathcal{L}^{\mathbb{G}}$ and $\mathcal{L}^{E}$ are initially empty. Then $\mathcal{O}$ calls $(e_1,\ldots,e_u) \leftarrow_{\$} \textsc{Encode}(G_1,\ldots,G_u)$ to determine encodings for all group elements $G_1,\ldots,G_u$ and starts the generic group algorithm on input $\mathcal{R}(e_1,\ldots,e_u,C')$. The reduction $\mathcal{R}^{\mathcal{O}}$ may now submit queries of the form $(e,e',\circ) \in E \times E \times \{\cdot,\div\}$ to the generic group oracle $\mathcal{O}$. In the sequel we will restrict $\mathcal{R}$ to issue only queries $(e,e',\circ)$ to $\mathcal{O}$ such that $e,e' \in \mathcal{L}^{E}$. It determines the smallest indices i and j with $e = e_i$ and $e' = e_j$ by calling $(i,j) = \textsc{GetIdx}(e,e')$. Then it computes $\mathcal{L}^{\mathbb{G}}_i \circ \mathcal{L}^{\mathbb{G}}_j$ and returns the encoding $\textsc{Encode}(\mathcal{L}^{\mathbb{G}}_i \circ \mathcal{L}^{\mathbb{G}}_j)$. Furthemore, we require that $\mathcal{R}$ only outputs encodings $\phi(S_i)$ such that $\phi(S_i) \in \mathcal{L}^{E}$.

Remark 30. We note that the above restrictions are without loss of generality. To explain this, recall that the assignment between group elements and encodings is random. An alternative

[2] Recall that the same encoding may occur multiple times in $\mathcal{L}^{E}$.

[3] Note that GetIdxmay receive only encodings $e_1,\ldots,e_w$ which are already contained in $\mathcal{L}^{E}$, as otherwise the behavior of GetIdx is undefined. We will make sure that this is always the case.

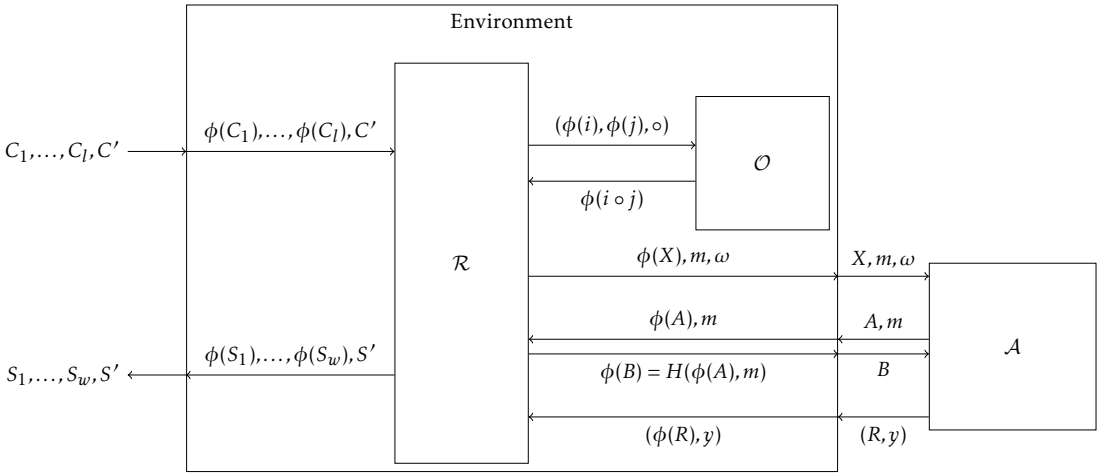


Figure 3.2: An example of the interaction between a generic reduction $\mathcal{R}$ and a non-generic adversary $\mathcal{A}$ against the unforgeability of Schnorr signatures. All group elements – such as the challenge input and the signature output by $\mathcal{A}$ – are encoded by the environment before being passed to $\mathcal{R}$. In the other direction, encodings of group elements output by $\mathcal{R}$ – such as the public key that is the input of $\mathcal{A}$and the solution output by $\mathcal{R}$ – are decoded before being passed to the outside world.

implementation $\mathcal{O}'$ of $\mathcal{O}$ could, given an encoding $e \notin \mathcal{L}^E$, assign a random group element $G \leftarrow_\$ \mathbb{G} \setminus \mathcal{L}^{\mathbb{G}}$ to e by appending G to $\mathcal{L}^{\mathbb{G}}$ and e to $\mathcal{L}^E$, in which case $\mathcal{R}$ would obtain an encoding of an independent, new group element. Of course $\mathcal{R}$ can simulate this behavior easily when interacting with $\mathcal{O}$, too.

Generic Reductions. Recall that a (fully black-box [RTV04]) reduction from problem Π to problem Σ is an efficient algorithm $\mathcal{R}$ that solves Π, having black-box access to an algorithm $\mathcal{A}$ solving Σ. In the sequel we consider reductions $\mathcal{R}^{\mathcal{A},\mathcal{O}}$ having black-box access to an algorithm $\mathcal{A}$ as well as to a generic group oracle $\mathcal{O}$. A generic reduction receives as input a challenge $C = (\phi(C_1),\ldots,\phi(C_u),C') \in \mathbb{G}^u \times \{0,1\}^*$ consisting of u encoded group elements and a bit-string C'. The reduction $\mathcal{R}$ may perform computations on encoded group elements, by invoking a generic group oracle $\mathcal{O}$ as described above, and interacts with algorithm $\mathcal{A}$ to compute a solution $S = (\phi(S_1),\ldots,\phi(S_w),S') \in \mathbb{G}^w \times \{0,1\}^*$, which again may consist of encoded group elements $\phi(S_1),\ldots,\phi(S_w)$ and a bit-string $S' \in \{0,1\}^*$.

We stress that the adversary $\mathcal{A}$ does not necessarily have to be a generic algorithm. It may not be immediately obvious that a generic reduction can make use of a non-generic adversary, considering that $\mathcal{A}$ might expect a particular encoding of the group elements. However, this is indeed possible. In particular, most reductions in security proofs for cryptosystems that are based on algebraic groups (e.g. [PS96; BB04; Wat05], to name a few well-known examples) are independent of a particular group representation, and thus generic.

Recall that $\mathcal{R}$ is fully black-box, i.e., $\mathcal{A}$ is external to $\mathcal{R}$. Thus, the environment in which the reduction runs can easily translate between the two encodings. Consider as an example the reduction shown in Figure 3.2 that interacts with a non-generic adversary $\mathcal{A}$. We stress that the

actual algorithm solving the problem Π, which is a composition of $\mathcal{R}$ and $\mathcal{A}$ is therefore *not* generic.

3.4 Unconditional Tightness Bound for Generic Reductions

In this section, we investigate the possibility of finding a tight *generic* reduction $\mathcal{R}$ that reduces a representation-invariant computational problem Π to breaking the UUF-NMA-security of the Schnorr signature scheme. Our results in this direction are negative, showing that it is impossible to find a tight generic reduction from any non-interactive representation-invariant computational problem.

We prove this main result using a meta-reduction technique. To recall, that means that we begin with the hypothesis that there exists a tight generic reduction $\mathcal{R}$ from some hard non-interactive problem Π to the UUF-NMA security of Schnorr signatures using only black-box access to the attacker $\mathcal{A}$. Then we show that under this hypothesis there exists an efficient algorithm $\mathcal{M}$, a meta-reduction, which efficiently solves Π. This implies that the hypothesis is false. The meta-reduction $\mathcal{M} = \mathcal{M}^{\mathcal{R}}$ uses $\mathcal{R}$ as a subroutine, and efficiently *simulates* the forger $\mathcal{A}$ for the reduction $\mathcal{R}$.

To show that the simulation of the attacker is correct, it is necessary to specify a concrete attacker that is being simulated. For this purpose we specify an explicit *inefficient* adversary. Next we describe how the adversary can be efficiently simulated by the meta-reduction. To do so, we make use of a novel simulation strategy that involves reprogramming the group representation. To showcase and explain this novel simulation strategy we first consider only a very simple class of reductions in Section 3.4.1. This restricted class of reductions allows for a much easier proof and a clearer explanation of our strategy. In Section 3.4.2 we finally apply the strategy to general case in order to achieve our main result.

3.4.1 Vanilla Reductions

We begin with considering a very simple class of reduction that we call *vanilla reductions*. A vanilla reduction is a reduction that runs the UUF-NMA forger $\mathcal{A}$ exactly once (without restarting or rewinding) in order to solve the problem Π. This allows us to explain and analyze the new simulation technique. Later we turn to reductions that may execute $\mathcal{A}$ repeatedly, such as the known security proof from [PS96] based on the Forking Lemma.

3.4.1.1 An Inefficient Adversary $\mathcal{A}$

In this section we describe an inefficient adversary $\mathcal{A}$ that breaks the UUF-NMA-security of the Schnorr signature scheme. Recall that a black-box reduction $\mathcal{R}$ must work for any attacker $\mathcal{A}$. Thus, algorithm $\mathcal{R}^{\mathcal{A}}$ will solve the challenge problem Π, given black-box access to $\mathcal{A}$. The meta-reduction will be able to simulate this attacker *efficiently* for any generic reduction $\mathcal{R}$. We describe this attacker for comprehensibility, in order to make our meta-reduction more accessible to the reader.

1. The input of $\mathcal{A}$ is a Schnorr public-key pk, a message m, and random coins $\omega \in \{0,1\}^n$.

2. The forger $\mathcal{A}$ chooses q uniformly random group elements $R_1, \ldots, R_q \leftarrow_\$ \mathbb{G}$. (We make the assumption that $q \leq |\mathbb{G}|$.) Subsequently, the forger $\mathcal{A}$ queries the random oracle H on (R_i, m) for all $i \in [q]$. Let $c_i := \mathsf{H}(R_i, m) \in \mathbb{Z}_p$ be the corresponding answers.

3. Finally, the forger $\mathcal{A}$ chooses an index uniformly at random $\alpha \leftarrow_\$ [q]$, computes $y \in \mathbb{Z}_p$ which satisfies the equation $g^y = \mathsf{pk}^{c_\alpha} \cdot R_\alpha$, and outputs (R_α, y). For concreteness, we assume this computation is performed by exhaustive search over all $y \in \mathbb{Z}_p$ (recall that we consider an unbounded attacker here, we show later how to instantiate it efficiently).

Note that (R_α, y) is a valid signature for message m with respect to the public key pk. Thus, the forger $\mathcal{A}$ breaks the UUF-NMA-security of the Schnorr signatures with probability 1.

3.4.1.2 Main Result for Vanilla Reductions

Now we are ready to prove our main result for vanilla reductions.

Theorem 31. *Let $\Pi = (\mathcal{G}_\Pi, \mathcal{V}_\Pi)$ be a non-interactive representation-invariant computational problem with a challenge consisting of u group elements and let p be the group order. Suppose there exists a generic vanilla reduction $\mathcal{R}$ that $(\epsilon_\mathcal{R}, t_\mathcal{R})$-solves Π, having* one-time *black-box access to an attacker $\mathcal{A}$ that $(\epsilon_\mathcal{A}, t_\mathcal{A})$-breaks the* UUF-NMA*-security of Schnorr signatures with success probability $\epsilon_\mathcal{A} = 1$ by asking q random oracle queries. Then there exists an algorithm $\mathcal{M}$ with access to $\mathcal{R}$ that (ϵ, t)-solves Π with $\epsilon \geq \epsilon_\mathcal{R} - \frac{2(u+q+t_\mathcal{R})^2}{p}$ and $t \approx t_\mathcal{R}$.*

Remark 32. The values u, q, and $t_\mathcal{R}$ are polynomially bounded while p is exponential. Therefore, the theorem shows that the existence of a reduction $\mathcal{R}$ implies the existence of a meta-reduction $\mathcal{M}$, which solves Π with essentially the same success probability and running time. Thus, an efficient (and even *non*-tight) reduction $\mathcal{R}$ can only exist if there exists an efficient algorithm for Π, which means that Π cannot be hard.

Remark 33. Observe that Theorem 31 rules out reductions from nearly arbitrary non-interactive computational problems. At a first glance this might look contradictory, for instance there always exists a trivial reduction from the problem of forging Schnorr signatures to solving the same problem. However, as explained in Example 29, forging Schnorr-signatures is not a representation-invariant computational problem, therefore this is not a contradiction.

Proof. Assume that there exists a generic vanilla reduction $\mathcal{R} := \mathcal{R}^{\mathcal{O},\mathcal{A}}$ that $(\epsilon_\mathcal{R}, t_\mathcal{R})$-solves Π, when given access to a generic group oracle $\mathcal{O}$, and a forger $\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$, where the inputs to the forger are chosen by $\mathcal{R}$. Furthermore, the reduction $\mathcal{R}$ simulates the random oracle H for $\mathcal{A}$. We denote the simulation of H by $\mathcal{R}$ by $\mathcal{R}.\mathsf{H}$. We show how to build a meta-reduction $\mathcal{M}$ that has black-box access to $\mathcal{R}$ and solves the representation-invariant problem Π directly.

We describe $\mathcal{M}$ in a sequence of games, beginning with an *inefficient* implementation $\mathcal{M}_0$ of $\mathcal{M}$ and modify it gradually until we obtain an *efficient* implementation $\mathcal{M}_2$ of $\mathcal{M}$. We bound the probability with which any reduction $\mathcal{R}$ can distinguish each implementation $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$ for all $i \in \{1, 2\}$, which yields that $\mathcal{M}_2$ is an efficient algorithm that can use $\mathcal{R}$ to solve Π if $\mathcal{R}$ is tight. In what follows let X_i denote the event that $\mathcal{R}$ outputs a valid solution to the given problem instance $\hat{C}$ of Π in Game i.

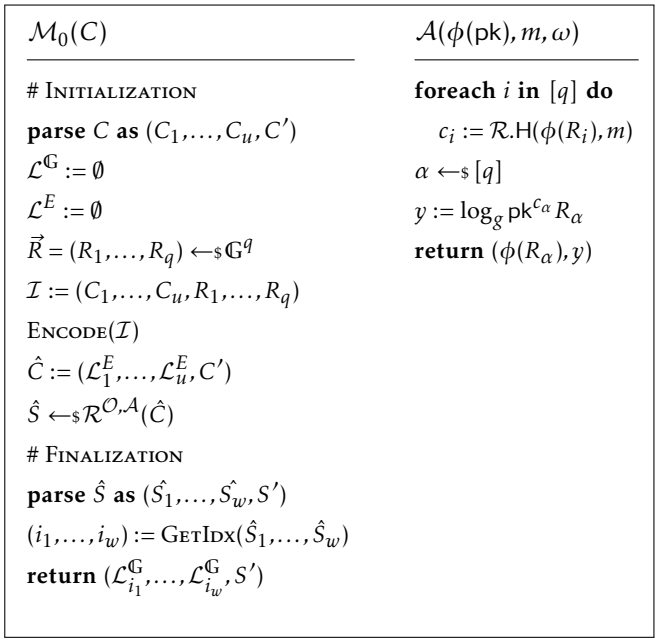


Figure 3.3: Implementation of $\mathcal{M}_0$.

Game 0. Our meta-reduction $\mathcal{M}_0$ is an algorithm for solving a representation-invariant computational problem Π, as defined in Section 3.3.3. That is, $\mathcal{M}_0$ takes as input an instance $C = (C_1,\ldots,C_u,C') \in \mathbb{G}^u \times \{0,1\}^*$, of the representation-invariant computational problem Π and outputs a candidate solution S. The reduction $\mathcal{R}$ is generic, i.e., a representation-independent algorithm for Π having black-box access to an attacker $\mathcal{A}$. Algorithm $\mathcal{M}_0$ runs the reduction $\mathcal{R}$ as a subroutine, by simulating the generic group oracle $\mathcal{O}$ and attacker $\mathcal{A}$ for $\mathcal{R}$. In order to provide the generic group oracle for $\mathcal{R}$, the meta-reduction $\mathcal{M}_0$ implements the following procedures (cf. Figure 3.3).

Initialization of $\mathcal{M}_0$: At the beginning of the game, $\mathcal{M}_0$ initializes two lists $\mathcal{L}^{\mathbb{G}} := \emptyset$ and $\mathcal{L}^{E} := \emptyset$, which are used to simulate the generic group oracle $\mathcal{O}$. Furthermore, $\mathcal{M}_0$ chooses $\vec{R} = (R_1,\ldots,R_q) \leftarrow_{\$} \mathbb{G}^q$ at random (these values will later be used by the simulated attacker $\mathcal{A}$), sets $\mathcal{I} := (C_1,\ldots,C_u,R_1,\ldots,R_q)$, and runs Encode($\mathcal{I}$) to assign encodings to these group elements. Then $\mathcal{M}_0$ invokes the reduction $\mathcal{R}$ on input $\hat{C} := (\mathcal{L}^E_1,\ldots,\mathcal{L}^E_u,C')$. Note that $\hat{C}$ is an encoded version of the challenge instance of Π received by $\mathcal{M}_0$. That is, we have $\hat{C} = (\phi(C_1),\ldots,\phi(C_u),C')$. Oracle queries of $\mathcal{R}$ are answered by $\mathcal{M}_0$ as follows.

Generic group oracle $\mathcal{O}(e,e',\circ)$: To simulate the generic group oracle, $\mathcal{M}_0$ implements procedures Encode and GetIdx as described in Section 3.3.4. Whenever $\mathcal{R}$ submits a query $(e,e',\circ) \in E \times E \times \{\cdot,\div\}$ to the generic group oracle $\mathcal{O}$, the meta-reduction determines the smallest indices i and j such that $e = \mathcal{L}^{\mathbb{G}}_i$ and $e' = \mathcal{L}^{\mathbb{G}}_j$ by calling $(i,j) =$ GetIdx(e,e'). Then it computes $\mathcal{L}^{\mathbb{G}}_i \circ \mathcal{L}^{\mathbb{G}}_j$ and returns Encode$(\mathcal{L}^{\mathbb{G}}_i \circ \mathcal{L}^{\mathbb{G}}_j)$.

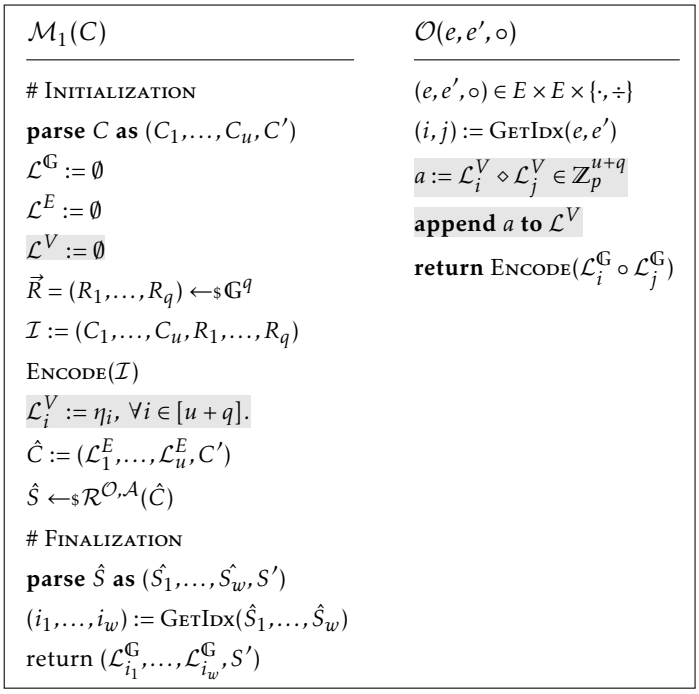


Figure 3.4: Meta-Reduction $\mathcal{M}_1$. Elements highlighted in gray show the differences to $\mathcal{M}_0$. All other procedures are identical to $\mathcal{M}_0$ and thus omitted.

The forger $\mathcal{A}(\phi(\mathsf{pk}),m,\omega)$: This procedure implements a simulation of the inefficient attacker $\mathcal{A}$ described in Section 3.4.1.1. It proceeds as follows. When $\mathcal{R}$ outputs $(\phi(\mathsf{pk}),m,\omega)$ to invoke an instance of $\mathcal{A}$, $\mathcal{A}$ queries the random oracle $\mathcal{R}.\mathsf{H}$ provided by $\mathcal{R}$ on $(\phi(R_i),m)$ for all $i \in [q]$, to determine $c_i = \mathsf{H}(\phi(R_i),m)$. Afterwards, $\mathcal{M}_0$ chooses an index $\alpha \leftarrow_{\$} [q]$ uniformly at random, computes the the discrete logarithm $y := \log_g \mathsf{pk}^{c_\alpha} R_\alpha$ by exhaustive search, and outputs $(\phi(R_\alpha),y)$. (This step is not efficient. We show in subsequent games how to implement this simulation efficiently.)

Finalization of $\mathcal{M}_0$: Eventually, the algorithm $\mathcal{R}$ outputs a solution $\hat{S} := (\hat{S_1},\dots,\hat{S_w},S') \in E^w \times \{0,1\}^*$. The algorithm $\mathcal{M}_0$ runs $(i_1,\dots,i_w) :=$ GetIdx$(\hat{S_1},\dots,\hat{S_w})$ to determine the indices of group elements $(\mathcal{L}_{i_1}^{\mathbb{G}},\dots,\mathcal{L}_{i_w}^{\mathbb{G}})$ corresponding to encodings $(\hat{S_1},\dots,\hat{S_w})$, and outputs $(\mathcal{L}_{i_1}^{\mathbb{G}},\dots,\mathcal{L}_{i_w}^{\mathbb{G}},S')$.

Analysis of $\mathcal{M}_0$. Note that $\mathcal{M}_0$ provides a perfect simulation of the oracle $\mathcal{O}$ and it also mimics the attacker from Section 3.4.1.1 perfectly. In particular, (R_α,y) is a valid forgery for message m and thus, $\mathcal{R}$ outputs a solution $\hat{S} = (\hat{S}_1,\dots,\hat{S}_w,S')$ to $\hat{C}$ with probability $\Pr[X_0] = \epsilon_{\mathcal{R}}$. Since Π is assumed to be representation-invariant, $S := (S_1,\dots,S_w,S')$ with $\hat{S}_i = \phi(S_i)$ for $i \in [w]$ is therefore a valid solution to C. Thus, $\mathcal{M}_0$ outputs a valid solution S to C with probability $\epsilon_{\mathcal{R}}$.

Game 1. In this game we introduce a meta-reduction $\mathcal{M}_1$, which essentially extends $\mathcal{M}_0$ with additional bookkeeping to record the sequence of group operations performed by $\mathcal{R}$. The

purpose of this intermediate game is to simplify our analysis of the final implementation $\mathcal{M}_2$. Meta-reduction $\mathcal{M}_1$ proceeds identical to $\mathcal{M}_0$, except for a few differences (cf. Figure 3.4).

Initialization of $\mathcal{M}_1$: The initialization is exactly as before, except that $\mathcal{M}_1$ maintains an additional list $\mathcal{L}^V$ of elements of $\mathbb{Z}_p^{u+q}$. Let $\mathcal{L}_i^V$ denote the i-th entry of $\mathcal{L}^V$.

List $\mathcal{L}^V$ is initialized with the $u+q$ canonical unit vectors in $\mathbb{Z}_p^{u+q}$. That is, let η_i denote the i-th canonical unit vector in $\mathbb{Z}_p^{u+q}$, i.e.,

$$\eta_1 := (1,0,\ldots,0), \eta_2 := (0,1,0,\ldots,0), \ldots, \eta_{u+q} := (0,\ldots,0,1).$$

Then $\mathcal{L}^V$is initialized such that $\mathcal{L}_i^V := \eta_i$ for all $i \in [u+q]$.

Generic Group Oracle $\mathcal{O}(e,e',\circ)$. In parallel to computing the group operation, the generic group oracle implemented by $\mathcal{M}_1$ also performs computations on vectors of $\mathcal{L}^V$. Given a query $(e,e',\circ) \in E \times E \times \{\cdot,\div\}$, the oracle $\mathcal{O}$ determines the smallest indices i and j such that $e = \mathcal{L}_i^{\mathbb{G}}$ and $e' = \mathcal{L}_j^{\mathbb{G}}$ by calling GetIdx. It computes $a := \mathcal{L}_i^V \diamond \mathcal{L}_j^V \in \mathbb{Z}_p^{u+q}$, where $\diamond := +$ if $\circ = \cdot$ and $\diamond := -$ if $\circ = \div$, and appends a to $\mathcal{L}^V$. Finally it returns Encode($\mathcal{L}_i^{\mathbb{G}} \circ \mathcal{L}_j^{\mathbb{G}}$).

Analysis of $\mathcal{M}_1$. Recall that the initial content $\mathcal{I}$ of $\mathcal{L}^{\mathbb{G}}$ is $\mathcal{I} = (C_1,\ldots,C_u,R_1,\ldots,R_q)$, and that $\mathcal{R}$ performs only group operations on $\mathcal{I}$. Thus, any group element $h \in \mathcal{L}^{\mathbb{G}}$ can be written as $h = \prod_{i=1}^u C_i^{a_i} \cdot \prod_{i=1}^q R_i^{a_{u+i}}$ where the vector $a = (a_1,\ldots,a_{u+q}) \in \mathbb{Z}_p^{u+q}$ is (essentially) determined by the sequence of queries issued by $\mathcal{R}$ to $\mathcal{O}$. For a vector $a \in \mathbb{Z}_p^{u+q}$ and a vector of group elements $V = (v_1,\ldots,v_{u+q}) \in \mathbb{G}^{u+q}$ let us write $\mathsf{Eval}(V,a)$ as a shorthand for $\mathsf{Eval}(V,a) := \prod_{i=1}^{u+q} v_i^{a_i}$ in the following. In particular, it holds that $\mathsf{Eval}(\mathcal{I},a) = \prod_{i=1}^u C_i^{a_i} \cdot \prod_{i=1}^q R_i^{a_{u+i}}$. The key motivation for the changes introduced in Game 1 is that now (by construction of $\mathcal{M}_1$) it holds that $\mathcal{L}_i^{\mathbb{G}} = \mathsf{Eval}(\mathcal{I},\mathcal{L}_i^V)$ for all $i \in \left[\left|\mathcal{L}^{\mathbb{G}}\right|\right]$. Thus, at any point in time during the execution of $\mathcal{R}$, the entire list $\mathcal{L}^{\mathbb{G}}$ of group elements can be recomputed from $\mathcal{L}^V$ and $\mathcal{I}$ by setting $\mathcal{L}_i^{\mathbb{G}} := \mathsf{Eval}(\mathcal{I},\mathcal{L}_i^V)$ for $i \in \left[\left|\mathcal{L}^V\right|\right]$. The reduction $\mathcal{R}$ is completely oblivious to this additional bookkeeping performed by $\mathcal{M}_1$, thus we have $\Pr[X_1] = \Pr[X_0]$.

Game 2. Note that the meta-reductions described in previous games were not efficient, because the simulation of the attacker in procedure $\mathcal{A}$ needed to compute a discrete logarithm by exhaustive search. In this final game, we construct a meta-reduction $\mathcal{M}_2$ that simulates $\mathcal{A}$ efficiently. $\mathcal{M}_2$ proceeds exactly like $\mathcal{M}_1$, except for the following (cf. Figure 3.5).

The forger $\mathcal{A}(\phi(\mathsf{pk}),m,\omega)$. When $\mathcal{R}$ outputs $(\phi(\mathsf{pk}),m,\omega)$ to invoke an instance of $\mathcal{A}$, the forger $\mathcal{A}$ queries the random oracle $\mathcal{R}.\mathsf{H}$ provided by $\mathcal{R}$ on $(\phi(R_i),m)$ for all $i \in [q]$, to determine $c_i = \mathsf{H}(\phi(R_i),m)$. Then it chooses an index $\alpha \leftarrow\!\!\$\, [q]$ uniformly at random, samples an element y uniformly at random from $\mathbb{Z}_p$, computes $R_\alpha^* := g^y \mathsf{pk}^{-c_\alpha}$, and re-computes the entire list $\mathcal{L}^{\mathbb{G}}$ using R_α^* instead of R_α.

More precisely, let $\mathcal{I}^* := (C_1,\ldots,C_u,R_1,\ldots,R_{\alpha-1},R_\alpha^*,R_{\alpha+1},\ldots,R_q)$. Observe that the vector $\mathcal{I}^*$ is identical to the initial contents $\mathcal{I}$ of $\mathcal{L}^{\mathbb{G}}$, with the difference that R_α is replaced by R_α^*. The list $\mathcal{L}^{\mathbb{G}}$is now recomputed from $\mathcal{L}^V$ and $\mathcal{I}^*$ by setting $\mathcal{L}_i^{\mathbb{G}} := \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}_i^V)$ for all $i \in \left[\left|\mathcal{L}^V\right|\right]$. Finally, $\mathcal{M}_2$ returns $(\phi(R_\alpha^*),y)$ to $\mathcal{R}$ as the forgery.

$\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$

---

foreach i **in** $[q]$ **do**
  $c_i = \mathcal{R}.\mathsf{H}(\phi(R_i), m)$
$\alpha \leftarrow\!\!\$\ [q]$
$y \leftarrow\!\!\$\ \mathbb{Z}_p \quad ; \quad R^*_\alpha := g^y \mathsf{pk}^{-c_\alpha}$
$\mathcal{I}^* := (C_1, \ldots, C_u, R_1, \ldots, R_{\alpha-1}, R^*_\alpha, R_{\alpha+1}, \ldots, R_q)$
for $j = 1, \ldots, \left|\mathcal{L}^{\mathbb{G}}\right|$ **do**
  $\mathcal{L}^{\mathbb{G}}_i := \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_i)$
return $(\phi(R^*_\alpha), y)$

Figure 3.5: Efficient simulation of attacker $\mathcal{A}$ by $\mathcal{M}_2$.

Analysis of $\mathcal{M}_2$. First note that $(\phi(R^*_\alpha), y)$ is a valid signature, since $\phi(R^*_\alpha)$ is the encoding of group element R^*_α satisfying the verification equation $g^y = \mathsf{pk}^{c_\alpha} \cdot R^*_\alpha$, where $c_\alpha = \mathsf{H}(\phi(R^*_\alpha), m)$. Next we claim that $\mathcal{R}$ is not able to distinguish $\mathcal{M}_2$ from $\mathcal{M}_1$, except for a negligibly small probability. To show this, observe that Game 2 and Game 1 are perfectly indistinguishable, if for all pairs of vectors $\mathcal{L}^V_i, \mathcal{L}^V_j \in \mathcal{L}^V$ it holds that $\mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_i) = \mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_j) \iff \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_i) = \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_j)$, because in this case $\mathcal{M}_2$ chooses identical encodings for two group elements $\mathcal{L}^{\mathbb{G}}_i, \mathcal{L}^{\mathbb{G}}_j \in \mathcal{L}^{\mathbb{G}}$ *if and only if* $\mathcal{M}_1$ chooses identical encodings. It remains to show that this happens with overwhelming probability. We state this in the following Lemma.

Lemma 34. *Let F denote the event that $\mathcal{R}$ computes vectors $\mathcal{L}^V_i, \mathcal{L}^V_j \in \mathcal{L}^V$ such that*

$$\mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_i) = \mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_j) \quad \wedge \quad \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_i) \neq \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_j) \tag{3.1}$$

or

$$\mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_i) \neq \mathsf{Eval}(\mathcal{I}, \mathcal{L}^V_j) \quad \wedge \quad \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_i) = \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}^V_j). \tag{3.2}$$

Then

$$\Pr[F] \leq 2(u + q + t_{\mathcal{R}})^2/p.$$

Before we prove Lemma 34, we first apply it to finish the proof of Theorem 31. By Lemma 34, algorithm $\mathcal{M}_2$ fails to simulate $\mathcal{M}_1$ with probability at most $2(u + q + t_{\mathcal{R}})^2/p$. Thus, we have $\Pr[X_2] \geq \Pr[X_1] - 2(u + q + t_{\mathcal{R}})^2/p$.

Note also that $\mathcal{M}_2$ provides an efficient simulation of adversary $\mathcal{A}$. The total running time of $\mathcal{M}_2$ is essentially of the running time of $\mathcal{R}$ plus some minor additional computations and bookkeeping. Furthermore, if $\mathcal{R}$ is able to $(\epsilon_{\mathcal{R}}, t_{\mathcal{R}})$-solve Π, then $\mathcal{M}_2$ is able to (ϵ, t)-solve Π with probability at least

$$\epsilon \geq \Pr[X_2] \geq \epsilon_{\mathcal{R}} - \frac{2(u + q + t_{\mathcal{R}})^2}{p}.$$

This concludes the proof of Theorem 31, subject to proving Lemma 34. □

It now remains to prove Lemma 34. The proof of this lemma is based on the observation that an algorithm that performs only a (polynomially) limited number of group operations in

an (exponential-size) generic group is very unlikely to find any "non-trivial relation" among random group elements. This technique was introduced in [Sho97] in a different setting, to analyze the complexity of algorithms for the discrete logarithm problem.

Proof of Lemma 34. We first introduce an alternative formulation of event F. Recall that the vectors $\mathcal{I}$ and $\mathcal{I}^*$ differ only in their α-th component. In the sequel let us write $\mathcal{I}_\alpha$ to denote the vector $\mathcal{I}$, but with its α-th component R_α set equal to $1 \in \mathbb{G}$. That is,

$$\mathcal{I}_\alpha := (R_1, \ldots, R_{\alpha-1}, 1, R_{\alpha+1}, \ldots, R_q, g_1, \ldots, g_u).$$

Then we have

$$\mathsf{Eval}(\mathcal{I}, \mathcal{L}_i^V) = \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V) \cdot R_\alpha^{\mathcal{L}_{i,\alpha}^V} \quad \text{and} \quad \mathsf{Eval}(\mathcal{I}^*, \mathcal{L}_i^V) = \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V) \cdot (R_\alpha^*)^{\mathcal{L}_{i,\alpha}^V}$$

where $\mathcal{L}_{i,\alpha}^V$ denotes the α-th component of vector $\mathcal{L}_i^V$. In particular, for any two vectors $\mathcal{L}_i^V, \mathcal{L}_j^V$ we have

$$\begin{aligned}\mathsf{Eval}(\mathcal{I}, \mathcal{L}_i^V) = \mathsf{Eval}(\mathcal{I}, \mathcal{L}_j^V) &\iff \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V) \cdot R_\alpha^{\mathcal{L}_{i,\alpha}^V} = \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_j^V) \cdot R_\alpha^{\mathcal{L}_{j,\alpha}^V} \\ &\iff \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V - \mathcal{L}_j^V) \cdot R_\alpha^{\mathcal{L}_{i,\alpha}^V - \mathcal{L}_{j,\alpha}^V} = 1.\end{aligned}$$

Thus, Equation 3.1 is equivalent to

$$\mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V - \mathcal{L}_j^V) \cdot R_\alpha^{\mathcal{L}_{i,\alpha}^V - \mathcal{L}_{j,\alpha}^V} = 1 \quad \wedge \quad \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V - \mathcal{L}_j^V) \cdot (R_\alpha^*)^{\mathcal{L}_{i,\alpha}^V - \mathcal{L}_{j,\alpha}^V} \neq 1. \tag{3.3}$$

If we take discrete logarithms to base $\gamma \in \mathbb{G}$, where γ is an arbitrary generator of $\mathbb{G}$, and define the degree-one polynomial $\Delta_{i,j,\alpha} \in \mathbb{Z}_p[X]$ as

$$\Delta_{i,j} := \log \mathsf{Eval}(\mathcal{I}_\alpha, \mathcal{L}_i^V - \mathcal{L}_j^V) + X \cdot \left(\mathcal{L}_{i,\alpha}^V - \mathcal{L}_{j,\alpha}^V\right),$$

then Equation 3.3 (and therefore also Equation 3.1) is in turn equivalent to

$$\Delta_{i,j}(\log R_\alpha) \equiv 0 \bmod p \qquad \wedge \qquad \Delta_{i,j}(\log R_\alpha^*) \not\equiv 0 \bmod p. \tag{3.4}$$

Similarly, Equation 3.2 is equivalent to

$$\Delta_{i,j}(\log R_\alpha) \not\equiv 0 \bmod p \qquad \wedge \qquad \Delta_{i,j}(\log R_\alpha^*) \equiv 0 \bmod p. \tag{3.5}$$

Thus, event F occurs if $\mathcal{R}$ computes vectors $\mathcal{L}_i^V, \mathcal{L}_j^V$ such that either Equation 3.4 or Equation 3.5 holds.

Failure Event F_1. Let F_1 denote the event that Equation 3.4 holds. Note that this can only happen if $\mathcal{R}$ performs a sequence of computations, such that there exist a pair $(i,j) \in \left[\left|\mathcal{L}^V\right|\right] \times \left[\left|\mathcal{L}^V\right|\right]$ such that the polynomial $\Delta_{i,j}$ is not the zero-polynomial in $\mathbb{Z}_p[X]$, but it holds that $\Delta_{i,j}(R_\alpha) \equiv 0 \bmod p$.

At the beginning of the game $\mathcal{R}$ receives only a random encoding $\phi(R_\alpha)$ of group element R_α. The only further information that $\mathcal{R}$ learns about R_α throughout the game is through equality or inequality of encodings. Since $\mathcal{R}$ runs in time $t_\mathcal{R}$, it can issue at most $t_\mathcal{R}$ oracle queries.

Thus, at the end of the game the list $\mathcal{L}^V$ contains at most $|\mathcal{L}^V| \leq t_{\mathcal{R}} + q + u$ entries. Each pair $(i,j) \in \left[\left|\mathcal{L}^V\right|\right]$ with $i \neq j$ defines a (possibly non-zero) polynomial $\Delta_{i,j}$. In total there are at most $(t_{\mathcal{R}} + q + u) \cdot (t_{\mathcal{R}} + q + u - 1) \leq (t_{\mathcal{R}} + q + u)^2$ such polynomials.

Since all polynomials have degree one, and $\log R_\alpha$ is uniformly distributed over $\mathbb{Z}_p$ (because R_α is uniformly random over $\mathbb{G}$), the probability that $\log R_\alpha$ is a root of any of these polynomials is upper bounded by

$$\Pr[F_1] \leq \frac{(u + q + t_{\mathcal{R}})^2}{p}.$$

Failure Event F_2. Let F_2 denote the event that Equation 3.5 holds. Since $\log R^*_\alpha$ is uniformly distributed over $\mathbb{Z}_p$ (because we have defined $R^*_\alpha := g^y \mathsf{pk}^{-c}$ for uniformly $y \leftarrow\!\!\$\, \mathbb{Z}_p$), with similar arguments as before we have

$$\Pr[F_2] \leq \frac{(u + q + t_{\mathcal{R}})^2}{p}.$$

Bounding $\Pr[F]$. Since $F = F_1 \cup F_2$ we have

$$\Pr[F] \leq \Pr[F_1] + \Pr[F_2] \leq \frac{2(u + q + t_{\mathcal{R}})^2}{p}.$$

□

3.4.2 Multi-Instance Reductions

Now we turn to considering multi-instance reductions, which may run multiple sequential executions of the signature forger $\mathcal{A}$. This is the interesting case, in particular because the Forking-Lemma based security proof for Schnorr signatures by Pointcheval and Stern [PS96] is of this type.

Again we construct a meta-reduction with simulated adversary. The main difference to our single-instance adversary is that it does not succeed with probability 1, but tosses a biased coin that decides if it forges for the message or not. On the first glance this approach might seem to be of little value, because an adversary with a higher success probability should improve the success probability of the reduction. However, it was shown in [Seu12] that, once we consider a reduction that runs multiple sequential executions of this adversary, this approach allows to derive an optimal tightness bound.

In the following we assume that the reduction $\mathcal{R}$ executes n sequential instances of the same adversary $\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$, where the public key $\phi(\mathsf{pk})$, the message m, and the randomness ω of each instance are chosen by $\mathcal{R}$. Observe that the input to the adversary and the random oracle query/answers completely determine the behaviour of the adversary. Thus, any successive execution of an instance of $\mathcal{A}$ may be identical to a previous execution up to a certain point, where the response $c = \mathsf{H}(R, m)$ of the random oracle differs from a response $c' = \mathsf{H}(R, m)$ received by $\mathcal{A}$ in a previous execution. This point is called the *forking point* [Seu12].

3.4.2.1 A Family of Inefficient Adversaries $\mathcal{A}_{F,f}$

In this section, we describe a different inefficient adversary $\mathcal{A}$ against the UUF-NMA-security of the Schnorr signature scheme. In fact, we do not describe a single adversary but a family of adversaries from which the meta-reduction will choose one to simulate at random.

To define this family, we fix the following notations. The Bernoulli distribution of a parameter $\mu \in [0,1]$ is defined by Ber_μ, i.e., $\Pr_{\delta \leftarrow\$ \mathrm{Ber}_\mu}[\delta = 1] = \mu$ and $\Pr_{\delta \leftarrow\$ \mathrm{Ber}_\mu}[\delta = 0] = 1-\mu$. Let $\mathcal{Q} = \mathbb{G}\times\mathbb{Z}_p$ be the set of possible random oracle queries and answers. By $\mathbb{S}_i = \mathcal{Q}^i$ we denote the set of random oracle query sequences of length i and the set of all possible sequences is defined as $\mathbb{S} = \bigcup_{i=1}^{q} \mathbb{S}_i$. Consider now the set $\mathbb{F}$ of all functions $F : \{0,1\}^\ell \times \mathbb{G} \times \{0,1\}^\kappa \times \mathbb{S} \to \mathbb{G}$. And the set $\mathbb{E}$ of functions $f : \mathbb{G} \to \{0,1\}$ for which the following holds $\Pr[f(g) = 1 \,|\, g \leftarrow\$ \mathbb{G}] = \Pr\left[b = 1 \,\middle|\, b \leftarrow\$ \mathrm{Ber}_\mu\right]$. For each pair $(F, f) \in \mathbb{F} \times \mathbb{E}$ we define the adversary $\mathcal{A}_{F,f}$ as follows:

1. The input of $\mathcal{A}$ is a Schnorr public-key pk, a message m, and random coins $\omega \in \{0,1\}^\kappa$.

2. The forger $\mathcal{A}$ sets $\sigma := \perp$ and performs the following computations. For $i = 1, \ldots, q$ it computes $R_i := F(m, \mathsf{pk}, \omega, (R_1, c_1), \ldots, (R_{i-1}, c_{i-1}))$ and queries the random oracle H on (R_i, m), where $c_i := \mathsf{H}(R_i, m) \in \mathbb{Z}_p$ is the corresponding answers. If $\sigma = \perp$, then $\mathcal{A}_{F,f}$ sets $Z_i := \mathsf{pk}^{c_i} R_i$ and checks if $f(Z_i) = 1$. If this is the case, then $\mathcal{A}_{F,f}$ computes $y_i \in \mathbb{Z}_p$ satisfying the equation $g^{y_i} = R_i \cdot \mathsf{pk}^{c_i}$ by exhaustive search and sets $\sigma := (R_i, y_i)$. Otherwise, if $f(Z_i) = 0$, then it continues with the loop.

3. Finally, the forger $\mathcal{A}_{F,f}$ returns σ.

Note that (R_i, y_i) is a valid signature for message m with respect to the public key pk. Thus, the forger $\mathcal{A}_{F,f}$ breaks the UUF-NMA-security of the Schnorr signatures whenever $f(Z_i) = 1$ for at least one $i \in [q]$. This translates to a success probability of $\epsilon_{\mathcal{A}} = 1 - (1-\mu)^q$.

Observe that defining the adversaries as above ensures that, while different instances of the same adversary will behave identically as long as their input and the answers of the random oracle are the same, as soon as one of the inputs or one of the random oracle answers differ the behavior of two instances will be independent of one another from that point onwards. As such, the behavior of these adversaries mimics closely the idea behind the forking lemma and it allows us to easily simulate the adversary in our meta-reduction below.

3.4.2.2 Main Result for Multi-Instance Reductions

In this section, we combine the approach of Seurin [Seu12] with our simulation of signature forgeries based on re-programming of the group representation, as introduced in Section 3.4.1.2. This allows to prove a nearly optimal unconditional tightness bound for all generic reductions and any representation-invariant computational problem Π.

Unfortunately, the combination of the elaborate techniques of Seurin [Seu12] with our approach yields a rather complex meta-reduction. We stress that we follow Seurin's work as closely as possible. The main difference lies in the way signature forgeries are computed, namely in our case by exploiting the properties of the generic group representation, instead of using an OMDL-oracle as in [Seu12].

The main difference between the meta-reduction described in this section and the one presented in Section 3.4.1.2 lies in the simulation of the random oracle queries issued by the adversary in different sequential executions. In particular, the meta-reduction $\mathcal{M}$ simulates the oracles procedures Encode, GetIdx, and $\mathcal{O}$ exactly as before.

Theorem 35. *Let Π be a representation-invariant computational problem. Suppose there exists a generic reduction $\mathcal{R}^{\mathcal{O},\mathcal{A}_{F,f}}$ that $(\epsilon_{\mathcal{R}}, t_{\mathcal{R}})$-solves Π, having n-*time *black-box access to an attacker $\mathcal{A}_{F,f}$ that $(\epsilon_{\mathcal{A}}, t_{\mathcal{A}}, q)$-breaks the* UUF-NMA*-security of Schnorr signatures with success probability $\epsilon_{\mathcal{A}} = 1-(1-\mu)^q$ in time $t_{\mathcal{A}} \approx q$. Then there exists an algorithm $\mathcal{M}$ that (ϵ, t)-solves Π with*

$$\epsilon \geq \epsilon_{\mathcal{R}} - \frac{2n(u+nq+t_{\mathcal{R}})}{p} - \frac{n\ln\left((1-\epsilon_{\mathcal{A}})^{-1}\right)}{q(1-p^{-1/4})}.$$

Remark 36. Just as in Theorem 31, the term $2n(u+nq+t_{\mathcal{R}})/p$ in the bound on ϵ is negligible. However, now we have an additional term $n\ln((1-\epsilon_{\mathcal{A}})^{-1}/(q(1-p^{-1/4}))$. Note that this term is identical to the corresponding term from the main theorem of [Seu12]. Following [Seu12], we therefore conclude that any reduction $\mathcal{R}$ must have a security loss of $\Omega(q)$.

Proof. Suppose that there exists a generic reduction $\mathcal{R} := \mathcal{R}^{\mathcal{O},\mathcal{A}}$ that $(\epsilon_{\mathcal{R}}, t_{\mathcal{R}})$-solves Π, when given access to a generic group oracle $\mathcal{O}$ and to n instances of the same forger $\mathcal{A}_{F,f}$, where the inputs to each instance of the forger are chosen by $\mathcal{R}$. As before, the random oracle $\mathcal{R}.\mathsf{H}$ for $\mathcal{A}$ is provided by $\mathcal{R}$. We show how to construct a meta-reduction $\mathcal{M}$ that has black-box access to $\mathcal{R}$ and that solves the representation-invariant problem Π directly. Again we proceed in a sequence of games, and denote with $\mathcal{M}_i$ the implementation of algorithm $\mathcal{M}$ in Game i, and with X_i the event that $\mathcal{R}$ outputs a valid solution S to C in Game i. As in Section 3.4.1.2, we will bound the probability with which any efficient reduction $\mathcal{R}$ can distinguish each implementation $\mathcal{M}_i$ from $\mathcal{M}_{i-1}$ for all $i \in \{1,2,3\}$. We start with an inefficient implementation $\mathcal{M}_0$ of $\mathcal{M}$, and modify this implementation gradually until we obtain an efficient algorithm $\mathcal{M}_3$ that uses $\mathcal{R}$ to solve Π.

Game 0. $\mathcal{M}_0$ (cf. Figure 3.6) takes as input an instance $C = (C_1, \ldots, C_u, C') \in \mathbb{G}^u \times \{0,1\}^*$ of the representation-invariant computational problem Π and outputs a candidate solution S. It also maintains the encoding of the group using two lists $\mathcal{L}^{\mathbb{G}} \subseteq \mathbb{G}$ and $\mathcal{L}^{E} \subseteq E$. Our first instance $\mathcal{M}_0$ perfectly simulates one adversary chosen from the family of adversaries described above uniformly at random. The only difference between the real and the simulated adversary is that the meta-reduction does not fix the functions F, f at the beginning but instead defines them on the fly.

Initialization of $\mathcal{M}_0$. At the beginning of the game, $\mathcal{M}_0$ chooses $\vec{R} = (R_{1,1}, \ldots, R_{n,q}) \leftarrow_{\$} \mathbb{G}^{nq}$ at random (these are the values the function F will be lazily programmed to evaluate to), sets $\mathcal{I} := (C_1, \ldots, C_u, R_{1,1}, \ldots, R_{n,q})$, and runs Encode($\mathcal{I}$) to assign encodings to these group elements. Furthermore, $\mathcal{M}_0$ initializes lists $\mathcal{T}$, Γ_{good}, Γ_{bad}, and $\mathcal{D}$ as empty lists. Recall that $\mathcal{R}$ executes n sequential instances of the simulated adversary $\mathcal{A}$ and that depending on the input and the query/answer pairs to $\mathcal{R}.\mathsf{H}$, the successive execution might be identical to a certain point. The list $\mathcal{T}$ will be used to store the inputs and query answer pairs of each adversary to ensure consistency of F across adversary instances. Note further that the simulated adversary tosses a biased coin and decides whether it forges a signature or not. The lists Γ_{good} and Γ_{bad} are used to store these

$\mathcal{M}_0(C)$

Initialization
parse C **as** $(C_1,\ldots,C_u,C')$
$\mathcal{L}^{\mathbb{G}} := \emptyset$
$\mathcal{L}^{E} := \emptyset$
$\vec{R} = (R_{1,1},\ldots,R_{n,q}) \leftarrow_{\$} \mathbb{G}^{nq}$
$\mathcal{I} := (C_1,\ldots,C_u,R_{1,1},\ldots,R_{n,q})$
Encode($\mathcal{I}$)
$\mathcal{T} := \emptyset$
$\Gamma_{\text{good}} := \emptyset$
$\Gamma_{\text{bad}} := \emptyset$
$\mathcal{D} := \emptyset$
$j := 0$
$\hat{C} := (\mathcal{L}^E_1,\ldots,\mathcal{L}^E_u,C')$
$\hat{S} \leftarrow_{\$} \mathcal{R}^{\mathcal{O},\mathcal{A}}(\hat{C})$
Finalization
parse $\hat{S}$ **as** $(\hat{S_1},\ldots,\hat{S_w},S')$
$(i_1,\ldots,i_w) :=$ GetIdx$(\hat{S}_1,\ldots,\hat{S}_w)$
return $(\mathcal{L}^{\mathbb{G}}_{i_1},\ldots,\mathcal{L}^{\mathbb{G}}_{i_w},S')$

$\mathcal{A}(\phi(\mathsf{pk}),m,\omega)$

$j := j+1$
$i_{\mathsf{pk}} :=$ GetIdx$(\phi(\mathsf{pk}))$
$\tau := (\phi(\mathsf{pk}),m,\omega)$
$\sigma := \perp$
$i := 1$
BeforeFork$(m,\mathcal{L}^{\mathbb{G}}_{i_{\mathsf{pk}}})$
AfterFork$(m,\mathcal{L}^{\mathbb{G}}_{i_{\mathsf{pk}}})$
append τ **to** $\mathcal{T}$
return σ

BeforeFork(m,pk)

$k :=$ EvalF(τ)
while $k \neq \perp$ **do**
  $c_i := \mathcal{R}.\mathsf{H}(R_k,m)$
  append (k,c_i) **to** τ
  $k' :=$ EvalF(τ)
  if $\sigma = \perp$
    $Z_i := R_k\mathsf{pk}^{c_i}$
    Encode(Z_i)
    if $k' = \perp$
      Fork(Z_i,k,c_i)
      $k := k'$
    else if $\phi(Z_i) \in \Gamma_{\text{good}}$
      $\sigma :=$ Forge(R_k,Z_i)
  $i := i+1$

AfterFork(m,pk)

while $i \leq q$ **do**
  $c_i := \mathcal{R}.\mathsf{H}(\phi(R_{j,i}),m)$
  append $((j,i),c_i)$ **to** τ
  if $\sigma = \perp$
    $Z_i := R_{j,i}\mathsf{pk}^{c_i}$
    Encode(Z_i)
    if $\phi(Z_i) \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$
      Decide$(Z_i,(j,i),c_i)$
    if $\phi(Z_i) \in \Gamma_{\text{good}}$
      $\sigma :=$ Forge$(R_{j,i},Z_i)$
    $i := i+1$

EvalF(τ)

foreach τ' **in** $\mathcal{T}$ **do**
  if $\tau \prec \tau'$
    $(k,c) := \tau'_{|\tau|+1}$
    return k
return $\perp$

Fork(Z,k,c)

if $\phi(Z) \notin \Gamma_{\text{good}} \cup \Gamma_{\text{bad}}$
  Decide(Z,k,c)
if $\phi(Z) \in \Gamma_{\text{good}}$
  $\sigma :=$ Forge(R_k,Z)

Forge(R,Z)

foreach (Z',y') **in** $\mathcal{D}$ **do**
  if $Z' = Z$
    return $(\phi(R),y')$

Decide(Z,k,c)

$\delta_z \leftarrow_{\$} \mathrm{Ber}_\mu$
if $\delta_z = 0$
  $\Gamma_{\text{bad}} = \Gamma_{\text{bad}} \cup \{\phi(Z)\}$
else
  $\Gamma_{\text{good}} = \Gamma_{\text{good}} \cup \{\phi(Z)\}$
  $y :=$ DLog(Z,k,c)
  append (Z,y) **to** $\mathcal{D}$

DLog(Z,k,c)

foreach y **in** $\mathbb{Z}_p$ **do**
  if $g^y = Z$
    return y

Figure 3.6: Meta-Reduction $\mathcal{M}_0$.

decisions whether for a given Z, the simulated adversary $\mathcal{A}_{F,f}$ will forge a signature or not. Again, they are used to ensure consistency of f across adversary instances. Accordingly Γ_{good} contains exactly those elements for which $\mathcal{M}_0$ knows the discrete logarithms and Γ_{bad} contains exactly those elements for which it will never compute the discrete logarithms. Finally, $\mathcal{D}$ is used to store known discrete logarithms. Then, $\mathcal{M}_0$ runs a black-box simulation of the reduction $\mathcal{R}$ on input $\hat{C} := (\mathcal{L}_1^E, \ldots, \mathcal{L}_u^E, C')$. Note that $\hat{C}$ is an encoded version of the challenge instance of Π received by $\mathcal{M}_0$. That is, we have $\hat{C} = (\phi(C_1), \ldots, \phi(C_u), C')$. Oracle queries of $\mathcal{R} = \mathcal{R}^{\mathcal{O},\mathcal{S}'_\Pi,\mathcal{A}}$ are answered exactly as described in Section 3.4.1.2, with the difference being the forger that we describe in the following.

The forger $\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$. The simulation of the forger $\mathcal{A}$ is rather technical, because $\mathcal{M}_0$ has to provide a consistent simulation of the n sequential executions of $\mathcal{A}$. As already discussed at the beginning of this chapter, $\mathcal{M}_0$ has to emulate an identical behavior of $\mathcal{A}$ up to the forking point, or the reduction might loose its advantage. We split this algorithm up into several sup-procedures (see Figure 3.6). The main sub-procedures are BeforeFork and AfterFork, with the idea that $\mathcal{A}$ runs the code of BeforeFork if the forking point has not been reached yet and the simulation must be consistent with a previous execution. The second procedure, AfterFork describes how $\mathcal{M}_0$ simulates $\mathcal{A}$ after the forking point.

Now we proceed with the technical description of the main procedure of $\mathcal{A}$ and explain the sub-procedures in the following. When $\mathcal{R}$ outputs $(\phi(\mathsf{pk}), m, \omega)$ to invoke an instance of $\mathcal{A}_{F,f}$, then $\mathcal{M}_0$'s simulation of $\mathcal{A}_{F,f}$ initializes the list τ with its input $(\phi(\mathsf{pk}), m, \omega)$ and the forgery σ with $\perp$. These inputs are part of the function F and we need to store them in oder to ensure consistency with previous adversary instances.

The forger's first stage BeforeFork(pk, m). In this stage, the forger first tries to evaluate the function F on its input using EvalF. If no previous instance with the same input exists, the instance has already forked and BeforeFork immediately returns. If the instance has not yet forked from all other instances, i.e., if there exists a previous instance with the same input, it receives back the index k of the R to which F evaluates. In this case it proceeds to ask query $c_i = \mathcal{R}.\mathsf{H}(\phi(R_k), m)$ and appends (k, c_i) to τ. If it has not already forged a signature it then computes $Z_i := R_k \mathsf{pk}^{c_i}$. If the forking point has been reached, the adversary now forks from the previous instances as described in Fork. Otherwise, if $Z_i \in \Gamma_{\text{good}}$, then $\mathcal{A}_{F,f}$ forges a signature by calling Forge(R_k, Z_i). The algorithm will repeat the described process until the forking point is reached.

The forger's second stage AfterFork(pk, m). After the current instance has forked from all previous instances it proceeds as follows. Until exactly q random oracle queries have been asked, $\mathcal{A}_{F,f}$ queries $c_i := \mathcal{R}.\mathsf{H}(\phi(R_{j,i}), m)$ and appends $((j,i), c_i)$ to τ. If the adversary has not already forged a signature, it continues to compute $Z_i := R_{j,i} \mathsf{pk}^{c_i}$. If $\phi(Z_i)$ is neither in Γ_{good} nor in Γ_{bad}, the adversary decides in which set to put it by invoking Decide. If afterwards $\phi(Z_i)$ is in Γ_{good}, a signature is forged. The algorithm continues in this fashion until exactly q random oracle queries have been asked.

Handling the forking point $\textsc{Fork}(Z,k,c)$. When the simulation of $\mathcal{A}_{F,f}$ reaches the forking point, it checks whether $\phi(Z)$ is contained neither in Γ_{good} nor in Γ_{bad} and if this is the case, the simulation decides in which set to put it by invoking $\textsc{Decide}$. If $\phi(Z)$ is contained in Γ_{good}, i.e. if $\mathcal{M}$ already knows the discrete logarithm, the simulation produces a forgery.

Deciding whether to forge $\textsc{Decide}(Z,k,c)$. To decide whether Z belongs in Γ_{good} or Γ_{bad}, the simulation tosses a biased coin $\delta_z \leftarrow\!\!\$ \mathrm{Ber}_\mu$. If $\delta_z = 0$ then Z is added to Γ_{bad}. If $\delta_z = 1$ then Z is added to Γ_{good}, its discrete logarithm y is computed using $\textsc{DLog}$ and (Z,y) is appended to $\mathcal{D}$.

Computing the discrete logarithm $\textsc{DLog}(Z,k,c)$. Computation of the discrete logarithm is performed by exhaustively searching for a $y \in \mathbb{Z}_p$ satisfying $g^y = Z$.

Producing a forgery $\textsc{Forge}(R,Z)$. Actually producing a forgery is trivial, because forgeries will only be produced for $Z \in \Gamma_{\text{good}}$ and for each such Z, $\mathcal{D}$ already contains the discrete logarithm. Accordingly, a forgery is produced by finding the entry $(Z',y') \in \mathcal{D}$ such that $Z' = Z$ and returning (R,y')

Finalization of $\mathcal{M}_0$. Eventually, $\mathcal{R}$ outputs a solution $\hat{S} := (\hat{S_1},\ldots,\hat{S}_w,S') \in \hat{G}^w \times \{0,1\}^*$. Then $\mathcal{M}_0$ runs $(i_1,\ldots,i_w) := \textsc{GetIdx}(\hat{S_1},\ldots,\hat{S}_w)$ to determine the indices of group elements $(\mathcal{L}^{\mathbb{G}}_{i_1},\ldots,\mathcal{L}^{\mathbb{G}}_{i_w})$ corresponding to encodings $(\hat{S_1},\ldots,\hat{S}_w)$, and outputs $(\mathcal{L}^{\mathbb{G}}_{i_1},\ldots,\mathcal{L}^{\mathbb{G}}_{i_w},S')$.

Analysis of $\mathcal{M}_0$ Note that $\mathcal{M}_0$ provides a perfect simulation of the oracle $\mathcal{O}$ and it also mimics the inefficient attacker from Section 3.4.2.1 perfectly, the only difference being that F is chosen lazily. In particular, (R,y') is a valid forgery for message m and thus, $\mathcal{R}^{\mathcal{O},\mathcal{A}_{F,f}}$ outputs a solution $\hat{S} = (\hat{S}_1,\ldots,\hat{S}_w,S')$ to $\hat{C}$ with probability $\Pr[X_0] = \epsilon_{\mathcal{R}}$. Since Π is assumed to be representation-invariant, $S := (S_1,\ldots,S_w,S')$ is therefore a valid solution to C, where $\hat{S}_i = \phi(S_i)$ for $i \in [w]$. Thus $\mathcal{M}_0$ outputs a valid solution S to C with probability $\epsilon_{\mathcal{R}}$.

Game 1. In this game we introduce an implementation $\mathcal{M}_1$ which extends $\mathcal{M}_0$ with bookkeeping, exactly as in Game 1 from the proof of Theorem 31. See Figure 3.7. Briefly summarized, we introduce an additional list $\mathcal{L}^V \subseteq \mathbb{Z}_p^{u+nq}$ to record the sequence of operations performed by $\mathcal{A}$. Let η_i denote the i-th canonical unit vector in $\mathbb{Z}_p^{u+nq}$. Then this list is initialized as $\mathcal{L}^V_i = \eta_i$ for $i \in [u+nq]$. Whenever $\mathcal{R}$ asks to perform a computation $(\mathcal{L}^E_i,\mathcal{L}^E_j,\circ)$, then $\mathcal{M}_1$ proceeds as before, but additionally appends $a := \mathcal{L}^V_i + \mathcal{L}^V_j \in \mathbb{Z}_p^{u+nq}$ (if $\circ = \cdot$) or $\mathcal{L}^V_i - \mathcal{L}^V_j \in \mathbb{Z}_p^{u+nq}$ (if $\circ = \div$) to $\mathcal{L}^V$.

Furthermore, in order to keep list $\mathcal{L}^V$ consistent with $\mathcal{L}^{\mathbb{G}}$ (exactly as in in the proof of Theorem 31), we replace the generic group oracle $\mathcal{O}$ of $\mathcal{M}_0$ with the following procedure.

Generic group oracle $\mathcal{O}(e,e',\circ)$ Given a query $(e,e',\circ) \in E \times E \times \{\cdot,\div\}$, the oracle $\mathcal{O}$ determines the smallest indices i and j such that $e = e_i$ and $e' = e_j$ by calling $\textsc{GetIdx}$. It computes $a := \mathcal{L}^V_i \diamond \mathcal{L}^V_j \in \mathbb{Z}_p^{u+nq}$, where $\diamond := +$ if $\circ = \cdot$ and $\diamond := -$ if $\circ = \div$, and appends a to $\mathcal{L}^V$. Finally it returns $\textsc{Encode}(\mathcal{L}^{\mathbb{G}}_i \circ \mathcal{L}^{\mathbb{G}}_j)$.

Recall that the initial content $\mathcal{I}$ of $\mathcal{L}^{\mathbb{G}}$ is $\mathcal{I} = (C_1,\ldots,C_u,R_{1,1},\ldots,R_{n,q})$, and that $\mathcal{R}$ performs only group operations on $\mathcal{I}$. Now, by construction of $\mathcal{M}_1$, it holds that $\mathcal{L}^{\mathbb{G}}_i = \mathrm{Eval}(\mathcal{I},\mathcal{L}^V_i)$ for

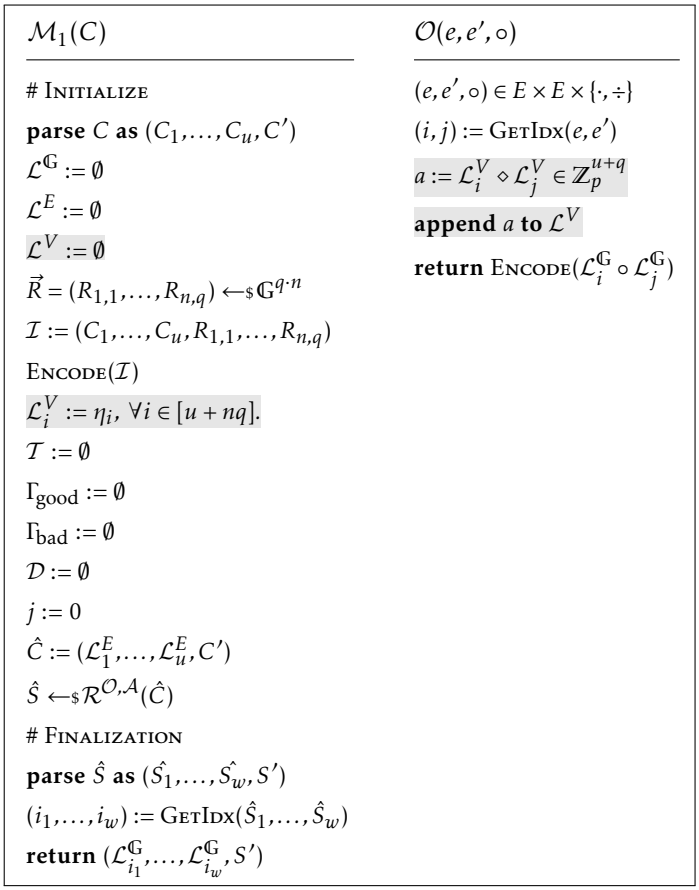


Figure 3.7: Extending $\mathcal{M}_0$ with additional bookkeeping yields $\mathcal{M}_1$. The elements highlighted in gray show the difference to $\mathcal{M}_0$. All procedures not shown are not changed.

all $i \in \left[\left|\mathcal{L}^{\mathbb{G}}\right|\right]$. Thus, at any point in time during the execution of $\mathcal{R}$, the entire list $\mathcal{L}^{\mathbb{G}}$ of group elements can be recomputed from $\mathcal{L}^{V}$ and $\mathcal{I}$ by setting $\mathcal{L}_i^{\mathbb{G}} := \mathsf{Eval}(\mathcal{I},\mathcal{L}_i^{V})$ for $i \in \left[\left|\mathcal{L}^{V}\right|\right]$.

Again this change is made to keep list $\mathcal{L}^{V}$ consistent with $\mathcal{L}^{\mathbb{G}}$, i.e., to ensure that $\mathcal{L}_i^{\mathbb{G}} = \mathsf{Eval}(\mathcal{I},\mathcal{L}_i^{V})$ for all $i \in \left[\left|\mathcal{L}^{\mathbb{G}}\right|\right]$, where $\mathcal{I} := (C_1,\dots,C_u,R_{1,1},\dots,R_{n,q})$. Clearly $\mathcal{R}$ is completely oblivious to this change, thus

$$\Pr[X_1] = \Pr[X_0]$$

Game 2. In this game we introduce an implementation $\mathcal{M}_2$ (cf. Figure 3.8) which works exactly as $\mathcal{M}_1$, except that it aborts when it would have to compute a new forgery at a forking point. That is, $\mathcal{M}_2$ aborts when it would have to forge in the case where it queried an R_i already asked by a previous instance of the adversary but received a different answer c_i. This step is important, because in the final implementation $\mathcal{M}_3$ we will not be able to simulate valid signatures if this happens.

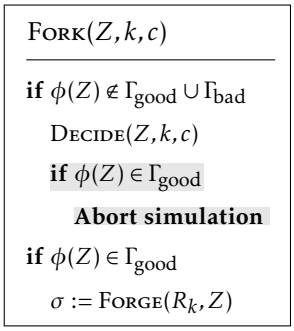


Figure 3.8: The difference between $\mathcal{M}_1$ and $\mathcal{M}_2$.

Fork(Z,k,c): If Fork is called on input $\phi(Z)$, such that $\phi(Z)$ is neither in Γ_{good} nor in Γ_{bad}, and the Decide places it in Γ_{good}, then $\mathcal{M}_2$ aborts.

Analysis of $\mathcal{M}_2$. We claim that $\mathcal{R}$ is not able to distinguish $\mathcal{M}_2$ from $\mathcal{M}_1$ with probability greater than $(n\ln\left((1-\epsilon_{\mathcal{A}})^{-1}\right))/q(1-p^{-1/4})$. To show this, observe that Game 2 and Game 1 are perfectly indistinguishable, as long as $\mathcal{M}_2$ does not abort in Fork. We use Lemma 4 of [Seu12] to bound the probability of an abort.

Lemma 37 (Based on Lemma 4 of [Seu12])**.** *The probability that $\mathcal{M}_2$ aborts in* Fork *is at most*

$$\frac{n\ln\left((1-\epsilon_{\mathcal{A}})^{-1}\right)}{q(1-p^{-1/4})}$$

We thus have

$$\Pr[X_2] \geq \Pr[X_1] - \Pr[F_1] \geq \Pr[X_1] - \frac{n\ln\left((1-\epsilon_{\mathcal{A}})^{-1}\right)}{q(1-p^{-1/4})}.$$

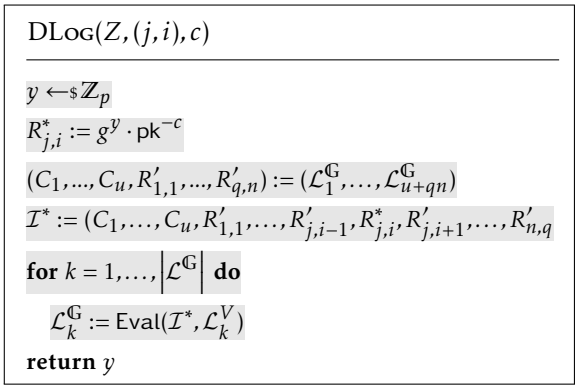


Figure 3.9: The difference between $\mathcal{M}_2$ and $\mathcal{M}_3$.

Game 3. Note that the meta-reductions described in previous games were not efficient, because the simulation of the attacker in procedure $\mathcal{A}$ needed to compute a discrete logarithm by exhaustive search. In this final game, we construct an efficient meta-reduction $\mathcal{M}_3$ that it identical to $\mathcal{M}_2$, with the difference that it simulates $\mathcal{A}$ efficiently. $\mathcal{M}_3$ proceeds exactly like $\mathcal{M}_2$, except for the following (cf. Figure 3.9).

DLog(Z,k,c): The DLog procedure chooses $y \leftarrow_\$ \mathbb{Z}_p$ uniformly random and computes

$$R^*_{j,i} := g^y \cdot \mathsf{pk}^{-c} \tag{3.6}$$

Then it reads the first $u+qn$ entries from $\mathcal{L}^{\mathbb{G}}$ as

$$(C_1,...,C_u,R'_{1,1},...,R'_{q,n}) := (\mathcal{L}^{\mathbb{G}}_1,\ldots,\mathcal{L}^{\mathbb{G}}_{u+qn}),$$

replaces $R_{j,i}$ with $R^*_{j,i}$ by setting

$$\mathcal{I}^* := (C_1,\ldots,C_u,R'_{1,1},\ldots,R'_{j,i-1},R^*_{i,j},R'_{j,i+1},\ldots,R'_{q,n}),$$

and finally re-computes the entire list $\mathcal{L}^{\mathbb{G}}$ from $\mathcal{L}^V$ by setting $\mathcal{L}^{\mathbb{G}}_a := \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}^V_a)$ for all $a \in \left[\left|\mathcal{L}^V\right|\right]$. Note that this implicitly defines Z as $Z := g^y$, due to Equation 3.6.

Note that meta-reduction $\mathcal{M}_3$ can be implemented efficiently, as it does not have to compute discrete logarithms. It remains to show that it is indistinguishable from $\mathcal{M}_2$ for $\mathcal{R}$ with all but negligible probablility.

Analysis of $\mathcal{M}_3$. First note that each σ with $\sigma \neq \perp$ output by $\mathcal{A}$ is a valid signature. Moreover, we claim that $\mathcal{R}$ is not able to distinguish $\mathcal{M}_3$ from $\mathcal{M}_2$, except for a negligibly small probability. To this end, we apply a lemma which is very similar to Lemma 34 from the proof of Theorem 31.

Lemma 38. *Let F_2 denote the event that $\mathcal{R}$ computes vectors $\mathcal{L}^V_a,\mathcal{L}^V_b \in \mathcal{L}^V$ such that*

$$\begin{aligned} &\mathsf{Eval}(\mathcal{I},\mathcal{L}^V_a) = \mathsf{Eval}(\mathcal{I},\mathcal{L}^V_b) \quad \wedge \quad \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}^V_a) \neq \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}^V_b) \\ &\qquad\qquad\qquad\qquad \textit{or} \\ &\mathsf{Eval}(\mathcal{I},\mathcal{L}^V_a) \neq \mathsf{Eval}(\mathcal{I},\mathcal{L}^V_b) \quad \wedge \quad \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}^V_a) = \mathsf{Eval}(\mathcal{I}^*,\mathcal{L}^V_b). \end{aligned}$$

Then

$$\Pr[F_2] \leq \frac{2n(u+nq+t_{\mathcal{R}})}{p}.$$

Before we sketch the proof of this lemma (which is very similar to the proof of Lemma 34), let us finish the proof of Theorem 31. Note that $\mathcal{M}_3$ is perfectly indistinguishable from $\mathcal{M}_2$, unless Event F occurs. Applying the above lemma, we thus obtain

$$\Pr[X_3] \geq \Pr[X_2] - \Pr[F_2] \geq \Pr[X_2] - \frac{2n(u+nq+t_{\mathcal{R}})}{p}.$$

Summing up, we thus obtain that

$$\epsilon \geq \epsilon_{\mathcal{R}} - \frac{2n(u+nq+t_{\mathcal{R}})}{p} - \frac{n\ln\left((1-\epsilon_{\mathcal{A}})^{-1}\right)}{q(1-p^{-1/4})}.$$

□

Proof Sketch for Lemma 38. The proof of Lemma 38 is almost identical to the proof of Lemma 34. The main difference is that we need to simulate many (up to n) signatures in the multi-instance case. This works well, with the same arguments as in the proof of Lemma 34, as long as we make sure that we do not need to re-assign the same encoding twice. (In particular because this would invalidate a signature previously computed by $\mathcal{A}$, and thus be easily noticeable for $\mathcal{R}$.)

By construction of $\mathcal{M}_3$, this can happen only if Fork receives as input a group element Z such that $\phi(Z) \in \Gamma_{\text{good}}$. Note that this is exactly when event F_1 occurs, in which case the game is aborted anyway, due to the changes introduced in Game 2.

Suppose that event F_1 does not occur. In this case we re-assign each encoding at most once, by replacing in list $\mathcal{L}^{\mathbb{G}}$ a uniformly distributed group element $R_{i,j}$ with another uniform group element $R^*_{i,j}$, and re-computing all group elements contained in $\mathcal{L}^{\mathbb{G}}$. Following Lemma 34, each replacement can be noticed by $\mathcal{R}$ with probability at most

$$\frac{2(u + nq + t_{\mathcal{R}})}{p},$$

where the term $u + nq$ (instead of $u + q$ as before) is due to the fact that in the multi-instance case $\mathcal{L}^{\mathbb{G}}$ is now initialized with $u + nq$ group elements. Since in total at most n encodings are re-assigned throughout the game, a union bound yields

$$\Pr[F_2] \leq \frac{2n(u + nq + t_{\mathcal{R}})}{p}.$$

□

3.5 On the Existence of Generic Reductions in the NPROM

In this section we apply our meta-reduction technique to a question orthogonal to the search for tight security proofs in the random oracle model. Namely, we investigate the possibility of finding any generic reduction $\mathcal{R}$ (even a non-tight one) that reduces a representation-invariant computational problem Π to breaking the UUF-NMA-security of the Schnorr signature scheme in the, weaker, *Non-Programmable* Random Oracle Model.

The standard security proof for Schnorr signatures in the ROM due to Pointcheval and Stern works by applying the forking lemma. That means the reduction to the underlying discrete logarithm problem essentially works as follows: It runs the adversary twice with the goal of receiving two forged signatures for the same message m and the same randomness r, but two different hash values $c := \mathsf{H}(g^r, m)$. This gives the reduction two values $y_1 := \mathsf{sk} \cdot c_1 + r$ and $y_2 := \mathsf{sk} \cdot c_2 + r$ and allows it to compute the secret key as

$$\frac{y_1 - y_2}{c_1 - c_2} = \frac{(\mathsf{sk} \cdot c_1 + r) - (\mathsf{sk} \cdot c_2 + r)}{c_1 - c_2} = \frac{\mathsf{sk} \cdot (c_1 - c_2)}{c_1 - c_2} = \mathsf{sk},$$

thereby computing the discrete logarithm of pk.

Clearly, this technique crucially relies on the fact that the reduction is able to *reprogram* the random oracle inbetween the two runs of the adversary, since otherwise c_1 and c_2 cannot be distinct. This is a standard technique for random oracle proofs, but it makes instantiating the random oracle very hard. Once the random oracle is instantiated with a real-world hash

function, such as SHA-2 or SHA-3, the function is fixed and cannot be reprogrammed. It is thus unclear if and how the reduction applies in this case.

Fischlin et al. [FLR+10] defined reductions in the non-programmable random oracle model (NPROM) by externalizing the random oracle to both the adversary *and* the reduction. [4] In such a scenario the reduction still observes the queries the adversary makes but cannot influence the replies. While the NPROM remains a highly idealized model, it matches much closer our intuition of "real" hash functions and instantiations through, say, SHA-3. And indeed, Fischlin et al. showed that the NPROM is a strictly weaker model than the standard ROM. It is therefore very interesting to ask whether a security proof for Schnorr signatures – even a non-tight one – might exist in this weaker model.

As in the sections before, our results are negative. We prove that it is impossible to find a generic reduction from any non-interactive representation-invariant computational problem in the NPROM.

3.5.1 An Inefficient adversary $\mathcal{A}$

The inefficient adversary $\mathcal{A}$ that breaks UUF-NMA-security of Schnorr signatures works as follows:

1. The input of $\mathcal{A}$ is a Schnorr public key $\mathsf{pk} \in \mathbb{G}$, a message $m \in \{0,1\}^\ell$, and random coins $\omega \in \{0,1\}^n$.
2. The forger $\mathcal{A}$ chooses a uniformly random $R \leftarrow_\$ \mathbb{G}$, queries the random oracle to computes $c := \mathsf{H}(R, m)$ and computes $Z := \mathsf{pk}^c R$.
3. Finally, the forger $\mathcal{A}$ uses exhaustive search to find $y \in \mathbb{Z}_p$ such that $Z = g^y$ and outputs (R, y).

Note that (R, y) is by definition of the Schnorr signature scheme always a valid signature for message m under public key pk. Thus, the forger described above breaks the UUF-NMA-security of Schnorr signatures with probability 1.

3.5.2 Main Result for Reductions in the NPROM

We will prove the following Theorem.

Theorem 39. *Let $\Pi = (\mathcal{G}_\Pi, \mathcal{V}_\Pi)$ be a representation-invariant non-interactive computational problem. Suppose there exists a generic reduction $\mathcal{R}$ that $(\epsilon_\mathcal{R}, t_\mathcal{R})$-solves Π, having n-*time *black-box access to an attacker $\mathcal{A}$ that $(\epsilon_\mathcal{A}, t_\mathcal{A}, q)$-breaks the* UUF-NMA*-security of Schnorr signatures in the non-programmable random oracle model with success probability $\epsilon_\mathcal{A} = 1$ in time $t_\mathcal{A} \approx 1$. Then there exists an algorithm $\mathcal{M}$ that (ϵ, t)-solves Π with $\epsilon \geq \epsilon_\mathcal{R}(1 - 2n(u + t_\mathcal{R})/p)$ and $t \approx t_\mathcal{R}$.*

Remark 40. The values n, u, and $t_\mathcal{R}$ are polynomially bounded while p is exponential. Therefore, the theorem shows that the existence of a reduction $\mathcal{R}$ implies the existence of a meta-reduction $\mathcal{M}$, which solves Π with essentially the same success probability and running time. Thus, an efficient (and even *non*-tight) reduction $\mathcal{R}$ can only exist if there exists an efficient algorithm for Π, which means that Π cannot be hard.

[4] The role of programmability was previously also investigated by Nielsen [Nie02], in a different setting.

$\mathcal{M}(C)$	$\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$
# Initialization	$e \leftarrow\!\!\$\, E$
parse C **as** $(C_1, \ldots, C_u, C')$	**if** $e \in \mathcal{L}^E$
$\mathcal{L}^{\mathbb{G}} := \emptyset$	**Abort simulation**
$\mathcal{L}^E := \emptyset$	$c := \mathcal{R}.\mathsf{H}(e, m)$
$\textsc{Encode}(C_1, \ldots, C_u)$	$y \leftarrow\!\!\$\, \mathbb{Z}_p$
$\hat{C} := (\mathcal{L}_1^E, \ldots, \mathcal{L}_u^E, C')$	$i := \textsc{GetIdx}(\phi(\mathsf{pk}))$
$\hat{S} \leftarrow\!\!\$\, \mathcal{R}^{\mathcal{O},\mathcal{A}}(\hat{C})$	$R := g^y \cdot (\mathcal{L}_i^{\mathbb{G}})^{-c}$
# Finalization	**if** $R \in \mathcal{L}^{\mathbb{G}}$
parse $\hat{S}$ **as** $(\hat{S_1}, \ldots, \hat{S_w}, S')$	**Abort simulation**
$(i_1, \ldots, i_w) := \textsc{GetIdx}(\hat{S}_1, \ldots, \hat{S}_w)$	**append** e **to** $\mathcal{L}^E$
return $(\mathcal{L}_{i_1}^{\mathbb{G}}, \ldots, \mathcal{L}_{i_w}^{\mathbb{G}}, S')$	**append** R **to** $\mathcal{L}^{\mathbb{G}}$
	return (e, y)

Figure 3.10: Implementation of $\mathcal{M}$.

Proof. Assume that there exists a generic reduction $\mathcal{R} := \mathcal{R}^{\mathcal{O},\mathcal{A}}$ that $(\epsilon_{\mathcal{R}}, t_r dv)$-solves Π when given access to a generic group oracle $\mathcal{O}$, and a forger $\mathcal{A}(\phi(\mathsf{pk}), m, \omega)$, where the inputs to the forger are chosen by $\mathcal{R}$. Furthermore, the reduction $\mathcal{R}$ can observe all random oracle queries made by $\mathcal{A}$, however it cannot influence the responses. We show how to build a meta-reduction $\mathcal{M}$ that has black-box access to $\mathcal{R}$ and solves the representation-invariant problem Π directly.

Note, that we may assume without loss of generality that $\mathcal{R}$ will never invoke $\mathcal{A}$ on the same input twice. This is because $\mathcal{R}$ cannot influence the random oracle responses and therefore the "forking" point for two instantiations must already be in the initial inputs. This makes things much simpler, as we do not have to ensure consistency between different instances of the adversary.

Meta-reduction $\mathcal{M}$. At the beginning of the game, $\mathcal{M}$ receives a challenge $C = (C_1, \ldots, C_u, C')$. It initializes the lists $\mathcal{L}^{\mathbb{G}} := \emptyset$ and $\mathcal{L}^E := \emptyset$ and determines encodings by running $(\phi(C_1), \ldots, \phi(C_u)) = \textsc{Encode}(C_1, \ldots, C_u)$. Then it invokes $\mathcal{R}^{\mathcal{A},\mathcal{O}}(\phi(C_1), \ldots, \phi(C_u), C')$. The oracle $\mathcal{O}$ is simulated exactly as in previous proofs (Refer to Figure 3.1 on page 48).

Whenever $\mathcal{R}$ outputs $(\phi(\mathsf{pk}), m, \omega)$ to invoke an instance of $\mathcal{A}$, $\mathcal{M}$ proceeds as follows. It chooses a random encoding $e \leftarrow\!\!\$\, E$, and and aborts if $e \in \mathcal{L}^E$. Then it queries the random oracle provided by $\mathcal{R}$ to compute $c := \mathsf{H}.\mathcal{R}(e, m)$, chooses $y \leftarrow\!\!\$\, \mathbb{Z}_p$, calls $i := \textsc{GetIdx}(\phi(\mathsf{pk}))$, and computes $R := g^y \cdot (\mathcal{L}_i^{\mathbb{G}})^{-c}$. It aborts if $R \in \mathcal{L}^{\mathbb{G}}$. Finally $\mathcal{M}$ appends e to $\mathcal{L}^E$ and R to $\mathcal{L}^{\mathbb{G}}$ and outputs (e, y) as a forgery.

Eventually, the algorithm $\mathcal{R}$ outputs a solution $\hat{S} := (\hat{S_1}, \ldots, \hat{S}_w, S') \in E^w \times \{0,1\}^*$. The algorithm $\mathcal{M}$ runs $(i_1, \ldots, i_w) := \textsc{GetIdx}(\hat{S_1}, \ldots, \hat{S}_w)$ to determine the indices of group elements $(\mathcal{L}_{i_1}^{\mathbb{G}}, \ldots, \mathcal{L}_{i_w}^{\mathbb{G}})$ corresponding to encodings $(\hat{S_1}, \ldots, \hat{S}_w)$, and outputs $(\mathcal{L}_{i_1}^{\mathbb{G}}, \ldots, \mathcal{L}_{i_w}^{\mathbb{G}}, S')$.

Analysis of $\mathcal{M}$. Note that $\mathcal{M}$ provides a perfect simulation of the oracle $\mathcal{O}$. Further, it mimics the attacker described in Section 3.5.1 perfectly unless it aborts while attempting to simulate a forger. We define two failure events. Let F_1 denote the event that while computing the forgery it holds that $e \leftarrow\!\!\$\, E$. Similarly, let F_2 denote the event that while computing the forgery it holds that $R \in \mathcal{L}^{\mathbb{G}}$. It holds that, if neither F_1 nor F_2 occur, $\mathcal{R}$ does not abort and (R, y) is always a valid forgery for message m and thus, $\mathcal{R}$ outputs a solution $\hat{S} = (\hat{S}_1, \ldots, \hat{S}_w, S')$ to $\hat{C}$ with probability $\epsilon_{\mathcal{R}}$. Since Π is assumed to be representation-invariant, $S := (S_1, \ldots, S_w, S')$ with $\hat{S}_i = \phi(S_i)$ for $i \in [w]$ is therefore a valid solution to C. Thus, the success probability of $\mathcal{M}$ is at least $\epsilon \geq \epsilon_{\mathcal{R}} \cdot (1 - \Pr[F_1 \cup F_2])$.

Since the n encodings chosen while simulating the forger $\mathcal{A}$ are chosen uniformly at random, and we have $\left|\mathcal{L}^E\right| \leq u + t_{\mathcal{R}}$ at all times, we can bound the probability that F_1 occurs using a union bound as $\Pr[F_1] \leq n \cdot (u + t_{\mathcal{R}})/p$. Similarly, since $y \leftarrow\!\!\$\, \mathbb{Z}_p$ is uniformly random and therefore R is a uniformly random group element in each simulated forger, we have $\Pr[F_2] \leq n \cdot (u + t_{\mathcal{R}})/p$. Therefore, using another union bound, we get that $\Pr[F_1 \cup F_2] \leq \Pr[F_1] + \Pr[F_2] \leq 2n(u + t_{\mathcal{R}})/p$.

Thus, in conclusion we obtain that

$$\epsilon \geq \epsilon_{\mathcal{R}} \left(1 - \frac{2n(u + t_{\mathcal{R}})}{p}\right)$$

as claimed. □

Remark 41. The above result does not carry over to the standard model. The reason is that the adversary is not necessarily generic. In the standard model, however, the hash function must be evaluated locally by both the reduction and the adversary. Since they are using different encodings of the group elements, a signature that appears valid for the adversary is invalid from the point of view of the reduction with overwhelming probability.

One might attempt to rectify this by specifying different hash functions to adversary and reduction, i.e., specifying $\mathsf{H}(\phi(\cdot))$ as the hash function for the adversary. However, this fails for the simple reason that $\phi(\cdot)$ is not necessarily efficiently computable.

4 On Statistically Secure Approximate Obfuscation

4.1 Introduction

Constructing public-key cryptography (e.g., public-key encryption) from private-key cryptography (such as one-way functions) is one of the most fundamental questions in theoretical cryptography, going back to the seminal paper of Diffie and Hellman [DH76]. Diffie and Hellman suggested that *program obfuscators* with sufficiently strong security properties would allow to realize this transformation. A program obfuscator is a compiler that takes as input a program, and outputs another program with equivalent functionality, but which is harder to reverse engineer. Diffie and Hellman suggested to obfuscate the encryption circuit of a symmetric-key encryption scheme, and use the obfuscated program as a public key so as to obtain a public-key encryption scheme. An additional hint that obfuscation may be instrumental in solving this riddle was provided by Impagliazzo and Rudich [IR90; IR89], who proved that a transformation from symmetric to public-key must make *non-black-box* use of the underlying symmetric primitive. Indeed, program obfuscation is one of very few non-black-box techniques known in cryptography.

Modern research showed that obfuscators with the security guarantees required by the Diffie-Hellman transformation do not exist in general [HS07; BGI+01; BGI+12]. However, recent years have seen incredibly prolific study of weaker notions of obfuscation, following the introduction of a candidate *indistinguishability obfuscator* (iO) by Garg et al. [GGH+13]. The security guarantee of iO is that the obfuscation of two functionally equivalent circuits should result in indistinguishable output distributions. That is, reverse engineering could not detect which of two equivalent implementations had been the source of the obfuscated program. Sahai and Waters [SW14] showed that even this seemingly weak notion suffices for private-key to public-key transformation (via a clever construction that does not resemble the Diffie-Hellman suggestion).

One would have hoped that a weak notion such as iO may be realizable with *statistical* security, i.e., that reverse engineering (to the limited extent required by iO) will not be possible even to an attacker with unlimited computational power. The existence of such *statistical indistinguishability obfuscators* (siO) would resolve the question of constructing public key cryptography from one-way functions, as well as allow to construct one-way functions based on the hardness of NP [KMN+14]. Alas, Goldwasser and Rothblum [GR07; GR14] proved that siO cannot exist unless the polynomial hierarchy collapses (in particular that it implies that statistical zero knowledge proofs exist for all languages in NP, i.e., $\mathsf{NP} \subseteq \mathsf{SZK}$, and it is known that $\mathsf{SZK} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. Therefore $\mathsf{NP} \subseteq \mathsf{coAM}$ which implies the collapse of the polynomial hierarchy), which is considered quite unlikely in computational complexity, and at any rate way beyond the current understanding of complexity theory. This seems to put a damper on our hopes to

achieve statistically secure obfuscation.

However, the [GR07; GR14] negative result crucially relies on the *correctness* of the obfuscator. That is, it only rules out such obfuscators that perfectly preserve the functionality of the underlying primitive (at least with high probability over the coins of the obfuscator). In contrast, the symmetric to public key transformation can be made to work with only *approximate* correctness, i.e., a non-negligible correlation between the functionality of the input circuit and that of the output circuit (where the probability is taken over the randomness of the obfuscator and the input domain). The question of whether statistical approximate iO (saiO) exists was therefore the new destination in the quest for understanding obfuscation. Interestingly, it turns out that ruling out *computational* notions of iO in some idealized models also boils down to the question of whether saiO exists (see Section 4.1.2 below). The study of this notion is the objective of this chapter.

Our Results. We show that statistical approximate iO (saiO) does not exist if one-way functions exist (under the assumption that $\mathsf{NP} \not\subseteq \mathsf{AM} \cap \mathsf{coAM}$). Thus, in particular that saiO cannot be used for the transformation from symmetric to public-key cryptography. We show that if one-way functions exist, then any non-negligible correlation between the output of the obfuscator and the input program would imply an algorithm for unique SAT (USAT) in $\mathsf{BPP}^{\mathsf{GapSD}}$, i.e., an algorithm deciding the SAT problem for formulae with at most one satisfying assignment in bounded-error probabilistic polynomial time when given access to an oracle deciding the Gap Statistical Distance Problem. As SAT reduces to USAT via a randomized reduction [VV85], a slight adaption of a result of Mahmoody and Xiao [MX10] shows that this implies that SAT is in $\mathsf{AM} \cap \mathsf{coAM}$.

To complement our result, we observe that if one-way functions do not exist, then an average-case notion of saiO exists for any distribution. Specifically, for any efficiently sampleable distribution over circuits, there exists an saiO obfuscator whose correctness holds with high probability over the circuits in that distribution (inverting the order of quantifiers would imply a worst-case saiO).

A Study of Correlation Obfuscation. Our impossibility results extend beyond the case of saiO. In fact, the result applies even when the *security* of the obfuscator is approximate. Namely, when we are only guaranteed that the obfuscation of functionally equivalent circuits results in distributions that have mild statistical distance (as opposed to negligible). This motivated us to explore the properties of this new kind of obfuscators, which as far as we know have not been studied in the literature before.

We consider statistical approximate *correlation* obfuscation sacO. A sacO obfuscator is characterized by two parameters $\epsilon \in [0, 1/2)$ and $\delta \in [0, 1)$. The requirement is that correctness holds with probability $1 - \epsilon$ (with respect to the randomness of the obfuscator and a random choice of input), and that obfuscating two functionally equivalent circuits results in distributions with statistical distance at most δ. The case of negligible δ is exactly saiO, discussed above, and the case of $\epsilon = 0$ corresponds to perfect correctness.

We observe that our impossibility result degrades gracefully and holds so long as $2\epsilon + 3\delta < 1$. We found this state of affairs unsatisfactory, and tried to extend the result to hold for the entire parameter range. However, it turns out that sacO exists via an almost trivial construction

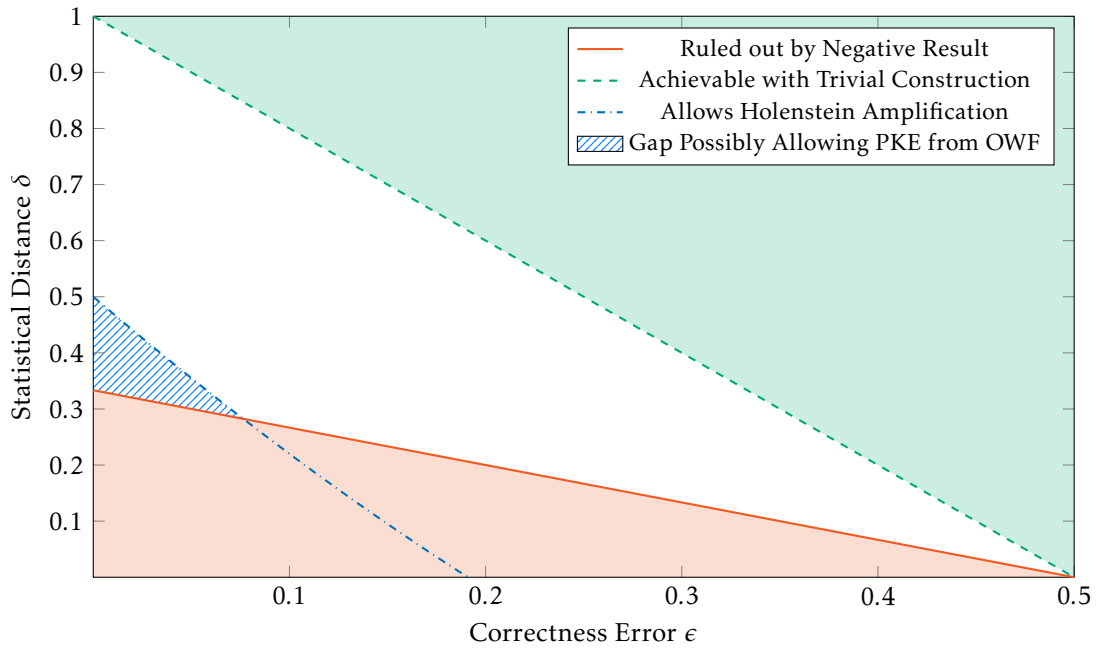


Figure 4.1: The graph gives an overview over the possible range of parameters for sacO. In the upper right are parameter regimes that can be achieved using the construction described in Section 4.4. In the lower left are the strong parameter regimes ruled out by our negative result in Section 4.3. The graph shows nicely the gap between the parameters that can be ruled out and those that can be used to construct public key encryption using the construction of Sahai and Waters as well as the amplification technique of Holenstein.

whenever $2\epsilon + \delta > 1$ (e.g., $\epsilon = \delta = 0.4$). We do not know if sacO exists in the intermediate parameter regime.

Lastly, we conduct a study of whether sacO is sufficient to construct public-key encryption from one-way functions. We present an amplified version of the Sahai-Waters construction using an amplification technique due to Holenstein. Interestingly, it appears that there is a region in the parameter domain that would allow to construct public-key encryption from one-way functions, but is not ruled out by our current technique. See Figure 4.1 for the landscape of sacO parameters. This leads to the intriguing open problem of closing the gap between the various parameter regimes, which is, however, beyond the scope of this thesis.

4.1.1 Our Techniques

Our starting point is the Goldwasser-Rothblum impossibility result [GR07; GR14]. Consider a statistical indistinguishability obfuscator such that for any pair of functionally equivalent circuits, the obfuscator generates statistically indistinguishable distributions, and in addition the output circuit of the obfuscator is always *functionally equivalent* to the input circuit (this can be relaxed to hold only with high probability over the random coins of the obfuscator). Goldwasser-Rothblum observe that an unsatisfiable SAT formula Ψ is functionally equivalent

to the all-zero function $\mathbf{0}$ and therefore the distributions produced by a siO obfuscator in both cases should be statistically indistinguishable. Slightly more formally, let $X[C]$ denote the distribution output by the obfuscator on input circuit C, then we get that $X[\Psi] \equiv X[\mathbf{0}]$, where $\equiv$ denotes statistical indistinguishability. On the contrary, if Ψ is a satisfiable formula, then it has a different functionality than $\mathbf{0}$ and therefore the support of $X[\Psi]$ and $X[\mathbf{0}]$ will be disjoint (and thus obviously not statistically indistinguishable). It follows that in order to solve SAT, it suffices to tell whether $X[\Psi]$ is close to $X[\mathbf{0}]$. It was shown by Sahai and Vadhan [SV97] that there exists an SZK protocol that takes two polynomial-time samplers, and decides whether they sample from distributions that are ϵ_1-statistically close or ϵ_2-statistically far, so long as $(\epsilon_2 - \epsilon_1)$ is large enough gap. The conclusion is that an siO obfuscator implies an SZK protocol for SAT which in turn implies that $\mathsf{NP} \subseteq \mathsf{SZK}$.

To sum up the core argument, to show that an siO obfuscator does not exists unless $\mathsf{NP} \subseteq \mathsf{SZK}$, Goldwasser-Rothblum built the formula-indexed distribution $X[\Psi]$ that samples an siO obfuscation of Ψ and has the properties that it is

(i) efficiently sampleable,

(ii) if Ψ is not satisfiable, then $X[\Psi]$ and $X[\mathbf{0}]$ are close, while

(iii) if Ψ not satisfiable, then $X[\Psi]$ and $X[\mathbf{0}]$ are far.

Allowing the obfuscator to have approximate correctness thwarts this approach completely. Hard SAT instances are obviously ones where the density of accepting inputs is sub-polynomial, since otherwise random sampling would yield a satisfying assignment with non-negligible probability. Therefore, satisfiable and unsatisfiable SAT formulae will have almost identical functionality. One could consider an saiO obfuscator that on any SAT formula that is not trivially satisfiable would just produce an obfuscation of $\mathbf{0}$. This means that $X[\Psi]$ will have the same distribution whether Ψ is satisfiable or not and thus, property (iii) is not satisfied anymore.

In order to overcome this issue, we construct a different distribution on formula-indexed circuits $C_X[k, \Psi]$ (where k is some uniformly random key k) such that if Ψ is not satisfiable, then $C_X[k, \Psi]$ and $C_X[k, \mathbf{0}]$ have the same functionality, and if Ψ is satisfiable, then $C_X[k, \Psi]$ and $C_X[k, \mathbf{0}]$ differ on a single point. Then, assuming one-way functions exist, we show that, although these two circuits differ on a single point only, when obfuscating $C_X[k, \Psi]$ the obfuscator has to produce a distribution that is statistically far from an obfuscation of $C_X[k, \mathbf{0}]$. To do this, we rely on the fact that the obfuscator itself is computationally efficient, and therefore it cannot break the hardness of one-way functions and derived cryptographic objects such as pseudorandom functions (PRFs) or *puncturable* PRFs (see below). This way, we construct a new formula-indexed distribution $X[\Psi]$ that satisfies properties (i), (ii) and (iii) as discussed above.

Puncturable PRFs were introduced simultaneously in [BW13; BGI14; KPT$^+$13] and were utilized as an essential building block for using indistinguishability obfuscation in [SW14]. A standard PRF is a function that can be efficiently computable using a key k, but is indistinguishable from a random function via oracle access. A puncturable PRF is a PRF where one can generate a *punctured key* $k\{x_0\}$ which allows to compute the PRF at all points except x_0, but the value at x_0 is still indistinguishable from uniform, even given the punctured key. Punctured PRFs can be constructed from any one-way function.

Based on a puncurable PRF and an saiO obfuscator O, we now construct a distribution on pairs of circuits (for now not indexed by a formula) such that the two circuits differ on a single

point only and yet, an saiO obfuscator will produce distributions that are far. Let k be a key for a puncturable PRF, let x_0 be a random point in the domain, let k^* be the key k punctured at x_0 and consider the function $f_{k^*,y}$ that outputs $\mathsf{PRF}(k^*,x) = \mathsf{PRF}(k,x)$ for all $x \neq x_0$, and outputs y on input x_0. Then by definition $f_{k^*,y}$ for a random y and $f_{k^*,y_0} = \mathsf{PRF}(k,\cdot)$ for $y_0 = \mathsf{PRF}(k,x_0)$ are identical in functionality except maybe at point x_0. However, using puncturing, we can guarantee that the distributions $\mathsf{O}(f_{k^*,y})$ and $\mathsf{O}(f_{k^*,y_0})$, where k, x_0, y are chosen uniformly at random are *statistically far*. To see this, it is enough to show that $\mathsf{O}(f_{k^*,y})$ and $\mathsf{O}(\mathsf{PRF}(k,\cdot))$ are statistically far since $f_{k^*,y_0} = \mathsf{PRF}(k,\cdot)$ and thus $\mathsf{O}(f_{k^*,y_0}) \equiv \mathsf{O}(\mathsf{PRF}(k,\cdot))$. Consider the predicate that checks whether $\mathsf{O}(\mathsf{PRF}(k,\cdot))(x_0) = \mathsf{PRF}(k,x_0)$. This predicate must have non-negligible bias towards holding true, and is efficiently checkable, which also implies that $\mathsf{O}(f_{k^*,y})(x_0) = f_{k^*,y}(x_0)$ holds true with noticeable bias, since otherwise we will have an efficient distinguisher from $f_{k^*,y_0} = \mathsf{PRF}(k,\cdot)$ in contradiction to the puncturable PRF security. Finally, since $y \neq y_0$ with high probability (assume for simplicity that the PRF and the obfuscator have long outputs and keys of half the size), this implies that $\mathsf{O}(f_{k^*,y})$ and $\mathsf{O}(f_{k^*,y_0})$ have noticeable statistical distance, since they will have noticeable probability mass on circuits that respect the functionality on x_0. Note that we used a *computational* argument, the security of punctured PRFs, to derive a *statistical* statement about the output distribution of the obfuscator.

We would like to use the aforementioned distributions to distinguish between satisfiable and unsatisfiable formulae. Let us restrict our attention to Unique-SAT formulae that are either unsatisfiable or have only one satisfying assignment. Unique-SAT is known to be NP-Hard via a randomized reduction [VV85], and combining results of Mahmoody and Xiao [MX10] and Bogdanov and Lee [BL13b] we show in Section 4.2.1 that if Unique-SAT is in $\mathsf{BPP}^{\mathsf{GapSD}}$, then SAT is in $\mathsf{AM} \cap \mathsf{coAM}$.

Let Ψ be a formula that has a unique satisfying assignment, then one can randomize the satisfying assignment (if it exists) to be uniformly distributed over the input space (e.g., by XORing all variables with a random string). Now, consider the function $f_{k,y,\Psi}$ defined s.t. $f_{k,y,\Psi}(x) = \mathsf{PRF}(k,x)$ if x does not satisfy Ψ, and $f_{k,y,\Psi}(x) = y$ otherwise. By definition, if Ψ is unsatisfiable then $f_{k,y,\Psi} = \mathsf{PRF}(k,\cdot)$ and if Ψ is satisfiable by some x_0 (which is uniformly distributed) then $f_{k,y,\Psi} = f_{k^*,y}$. Therefore $\mathsf{O}(f_{k,y,\Psi})$ is guaranteed to have a noticeable statistical distance in the case where Ψ is unsatisfiable (in which case it is close to $\mathsf{O}(f_{k,y,\mathbf{0}})$) and in the case where it is uniquely satisfiable (in which case it is far from $\mathsf{O}(f_{k,y,\mathbf{0}})$). This will allow us to produce an SZK protocol to distinguish the two possibilities.

In a World without OWFs. We recall that if OWFs do not exist then for any efficiently computable function f and with overwhelming probability over a y sampled from the output distribution of f, it is possible to efficiently sample (almost) uniformly (up to an arbitrarily small but fixed inverse polynomial error) from the set $f^{-1}(y) = \{x : f(x) = y\}$ [IL89]. Given an efficiently sampleable distribution over circuits, we can construct an average-case obfuscator for this family as follows. Let sampC be a sampler for this distribution of circuits and consider the function $f(r,x_1,\ldots,x_m)$ for a large polynomial m such that $f(r,x_1,\ldots,x_m) = (x_1,\ldots,x_m,C(x_1),\ldots,C(x_m))$, for $C = \mathsf{sampC}(r)$.

Now, to obfuscate a circuit C, sample $x_1,\ldots,x_m$ and compute $y_i = C(x_i)$. Then sample $(r,x_1,\ldots,x_m)$ from $f^{-1}(x_1,\ldots,x_m,y_1,\ldots,y_m)$ and finally output $C' = \mathsf{sampC}(r)$. This is clearly a perfect indistinguishability obfuscator (i.e., two circuits with the same functionality will produce identical distributions). It is also approximately correct on the average, because on average, if

two circuits agree on a randomly chosen set of points, then they will have a large agreement altogether.

We note that a similar and even simpler argument shows that if all efficiently computable functions are PAC learnable [Val84], even allowing membership queries, then saiO with perfect indistinguishability exists. A probably approximately correct (PAC) learner is an algorithm that with high probability outputs a hypothesis that approximately agrees with the function being learned, i.e., agrees on a large fraction of inputs. An saiO follows immediately by giving the learner (black-box) access to C, and outputting its hypothesis C' as the output of the obfuscator. In such case OWFs trivially do not exist.

The Landscape of Correlation Obfuscation. Extending our techniques to rule out sacO with $2\epsilon + 3\delta < 1$ follows from carefully analyzing the parameters in the proof outlined above (one can get $2\epsilon + 4\delta < 1$ by straightforward analysis, and the slight improvement comes from properly defining the random variables in the problem). We can show a trivial sacO obfuscator for $2\epsilon + \delta > 1$ as follows. Given an input circuit C, use random sampling to find the majority value of the truth table of C (if C is approximately balanced, then any value works). Then output the constant function taking the majority value with probability 2ϵ, and output C itself with probability $1 - 2\epsilon$. Correctness will hold with probability $1 - \epsilon$, since if C is output then correctness is perfect, and if the constant function is output then correctness is approximately $1/2$. The correlation between two functionally equivalent circuits is at least 2ϵ since the calculation of the majority value only depends on the truth table. We provide a more formal analysis in Section 4.4. It seems that such a trivial obfuscator cannot imply any non-trivial results.

We notice that a sacO obfuscator can be plugged into the Sahai-Waters construction, and would imply weak notions of security and correctness for the resulting public-key encryption scheme. Holenstein [Hol06] shows that, for some parameters, this weak notion can be amplified to standard security and correctness. Plugging in our parameters, we get that roughly when $\frac{1}{2} - 3\epsilon + 2\epsilon^2 > \delta$, sacO would imply symmetric to public key transformation using this method. This leaves a small region of parameters where sacO is not known to be impossible, and if it is possible it will imply highly non-trivial results. It is not clear whether other parameter regimes can also be useful, or whether our impossibility can be extended to rule out the entire useful regime. We refer to Figure 4.1 again for a visual characterization of the parameter regimes.

4.1.2 Consequences of Our Result

Our result strengthens previous negative results for proving the existence of iO in several ideal models. Previous works show that a construction of statistically secure (perfectly correct) iO in any of those ideal models implies the existence of saiO in the standard model. Actually, one can generalize these results to also hold for saiO. Combined with our result, we now yield that a construction of iO or saiO in these ideal models implies that $\mathsf{NP} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ or the non-existence of one-way functions.

This line of research was initiated by Canetti et al. [CKP15] who show that given a VBB obfuscator in the random oracle model, one can remove the random oracle at the cost of relaxing the correctness of the obfuscator. Pass and shelat [PS16] show an analogous result for VBB obfuscators in the ideal constant-degree encoding model, and Mahmoody, Mohammed, and Nematihaji [MMN16] show analogous results for the generic group model and the random

trapdoor permutation model. All these results transform a VBB obfuscator in an oracle world into an approximately correct VBB obfuscator in the standard model. They yield an impossibility result for VBB obfuscation in the ideal models, as approximately correct VBB is known not to exist, assuming trapdoor permutations, see [CKP15; BP13]. The crucial insight of Mahmoody et al. [MMN+16] is that all these oracle removal procedures are actually oblivious to the exact notion of obfuscation. The reason is that all proofs proceed by showing that the oracle-free obfuscation is as secure as the oracle-based obfuscation, i.e., the oracle-free obfuscated circuit can be simulated by an adversary in the oracle world, given the oracle-based obfuscated circuit. Therefore, if one has an iO obfuscator in any of the ideal models, via the oracle removal procedures, one obtains an saiO obfuscator in the standard model. Mahmoody et al. [MMN+16] conclude that, as an saiO obfuscator in the standard model allows to resolve the long-standing open problem of building public-key encryption from symmetric-key encryption, it seems very hard to construct such an object. In other words, their result rules out saiO assuming that building public-key encryption from symmetric-key encryption is impossible. Our result strengthens[1] their result by ruling out saiO based on the accepted complexity postulate that $\mathsf{NP} \not\subseteq \mathsf{AM} \cap \mathsf{coAM}$ and the fundamental assumption of cryptography that one-way functions exist. Therefore, based on the same assumptions, iO in all aforementioned idealized models cannot exist.

4.1.3 Open Problems

The main question that we leave open is the set of parameters for sacO that are useful and that are (im)possible. Note that it is desirable to have more positive results not only for sacO, but also for acO, the *computational* variant of sacO, in the spirit of Bitansky-Vaikuntanathan [BV16] who give an assumption-based transformations from aiO to standard iO. Even if sacO for useful parameters turns out to be impossible, it might still be easier to build acO for useful parameters and then use amplification rather than to build fully secure fully correct iO directly.

In particular, note that for a certain parameter range of sacO, we do not know of any impossibility results of building sacO in ideal models. The oracle removal procedures that we discuss in Section 4.1.2 maintain security and only weaken correctness. Therefore, a variant of the oracle removal procedures can also be proven for sacO (losing some amount of correctness). As not all useful parameters for sacO are ruled out by our results, one might aim for building sacO in an ideal model for these parameters. Note that one can use our result as a sanity check for any potential oracle construction: If the construction would also work for parameters that we rule out, then it is probably better to pursue a different approach.

Another direction for building useful statistical variants of iO is to relax the computational efficiency of the obfuscator in which case the distributions $X[\Psi]$ that we considered before are not efficiently sampleable anymore and thus, the argument described in Section 4.1.1 fails. Interestingly, Lin et al. [LPS+16] recently showed that such a notion of iO, which they call XiO, indeed has useful applications to transformations on functional encryption.

[1] Note that our result is only a "stronger" result in a moral sense, but not in a formal sense. While the non-existence of one-way function would allow us to build a reduction from public-key encryption to symmetric-key encryption (as in this case, both do not exist), it is not known that $\mathsf{NP} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ implies that we can build a public-key encryption scheme from a one-way function.

4.2 Preliminaries

4.2.1 Complexity Theory

We refer the reader to Goldreich's book [Gol08] for a detailed exposition of complexity theory. We now discuss a few object that are most relevant to our proof. We let SAT denote the set of all satisfiable boolean formulae in conjunctive normal form (CNF), we let USAT denote the set of CNF formulae that have exactly one satisfying assignment, and we let UNSAT denote the set of CNF formulae that have no satisfying assignment. Given a formula Ψ, deciding whether $\Psi \in \mathsf{SAT}$ is an NP-Complete problem. We recall that a *promise problem* $\Pi = (\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}})$ is a pair of disjoint subsets of $\{0,1\}^*$. Of particular interest to us is the *unique SAT* (promise) problem $\mathsf{UniqueSAT} = (\mathsf{USAT}, \mathsf{UNSAT})$. Total problems (a.k.a. languages) are a special case of promise problems, e.g. $(\mathsf{SAT}, \mathsf{UNSAT})$ is exactly the SAT problem. In such a case, it suffices to specify Π_{Yes} in order to completely define the problem.

We consider the notion of *randomized polynomial time Turing reductions* between problems. A *promise oracle* to a problem $\Pi = (\Pi_{\mathsf{Yes}}, \Pi_{\mathsf{No}})$, is one that always answers 1 on inputs in Π_{Yes} and always answers 0 on inputs in Π_{No}, but otherwise can answer arbitrarily, and even inconsistently between calls. We define the class BPP^{Π} as the class of problems solvable using a probabilistic polynomial time algorithm with access to a Π oracle. In other words, BPP^{Π} is the class of problems that are *reducible* to Π. One can verify that this class indeed composes, i.e. if $\widetilde{\Pi} \in \mathsf{BPP}^{\Pi}$ then $\mathsf{BPP}^{\widetilde{\Pi}} \subseteq \mathsf{BPP}^{\Pi}$. Valiant and Vazirani [VV85] showed that SAT is reducible to unique SAT.

Theorem 42 (Valiant-Vazirani)**.** $\mathsf{SAT} \in \mathsf{BPP}^{\mathsf{UniqueSAT}}$.

An additional promise problem which will be of interest to us is the GapSD problem, defined by Sahai and Vadhan [SV97]. This problem essentially captures the hardness of distinguishing between efficient samplers for statistically close distributions and ones for statistically far distributions. We recall that for a circuit C (which we regard as a sampler from a distribution), $C(\mathcal{U})$ denotes the distribution generated by running C on a random input.

Definition 43 (GapSD Problem)**.** *The problem* $\mathsf{GapSD} = (\mathsf{GapSD}_{\mathsf{Yes}}, \mathsf{GapSD}_{\mathsf{No}})$ *is defined as follows. Consider tuples of the form* $(C_0, C_1, v, 1^\ell)$*, where* C_0, C_1 *are circuits,* v *is a threshold value and* 1^ℓ *is a unary encoding of a probability gap. Define*

$$\mathsf{GapSD}_{\mathsf{Yes}} = \{(C_0, C_1, v, 1^\ell) : \mathsf{SD}(C_0(\mathcal{U}), C_1(\mathcal{U})) < v\} ,$$

and

$$\mathsf{GapSD}_{\mathsf{No}} = \{(C_0, C_1, v, 1^\ell) : \mathsf{SD}(C_0(\mathcal{U}), C_1(\mathcal{U})) > v + 1/\ell\} .$$

Combining results by Mahmoody and Xiao [MX10] and by Bogdanov and Lee [BL13b] as follows implies that $\mathsf{BPP}^{\mathsf{GapSD}}$ is contained in $\mathsf{AM} \cap \mathsf{coAM}$. In fact, by applying [MX10] we get that $\mathsf{BPP}^{\mathsf{SZK}} \in \mathsf{AM} \cap \mathsf{coAM}$, which is almost what we need. However, it is only known that $\mathsf{GapSD} \in \mathsf{SZK}$ under a somewhat weaker definition of the GapSD problem.

Theorem 44. $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$.

Proof. It follows from [BL13b, Theorem 9] that $\mathsf{GapSD} \in \mathsf{AM} \cap \mathsf{coAM}$. This means that both $(\mathsf{GapSD}_{\mathsf{Yes}}, \mathsf{GapSD}_{\mathsf{No}})$ and its complement $(\mathsf{GapSD}_{\mathsf{No}}, \mathsf{GapSD}_{\mathsf{Yes}})$ have AM protocols, say with

completeness 9/10 and soundness 1/10. Consider the protocol that takes $(C_0, C_1, \nu, 1^\ell)$ and does the following. First, execute the AM protocol for $(\mathsf{GapSD}_{\mathsf{Yes}}, \mathsf{GapSD}_{\mathsf{No}})$ on input $x_1 = (C_0, C_1, \nu + 1/(4\ell), 1^{4\ell})$. Then, execute the AM protocol for $(\mathsf{GapSD}_{\mathsf{No}}, \mathsf{GapSD}_{\mathsf{Yes}})$ (note the reverse order) on $x_2 = (C_0, C_1, \nu - 1/(2\ell), 1^{4\ell})$. Accept only if the two executions accepted. Now, assume that $\nu = \mathsf{SD}(C_0, C_1)$. Then it holds that $x_1 \in \mathsf{GapSD}_{\mathsf{Yes}}$ and $x_2 \in \mathsf{GapSD}_{\mathsf{No}}$ and therefore our new protocol accepts with probability at least 8/10. However, if $|\nu - \mathsf{SD}(C_0, C_1)| > 1/\ell$ then either $x_1 \in \mathsf{GapSD}_{\mathsf{No}}$ or $x_2 \in \mathsf{GapSD}_{\mathsf{Yes}}$ and therefore our new protocol accepts with probability at most 2/10. This means that our protocol is an AM protocol that, for any ϵ, can decide given (C_0, C_1), $1^{\lceil 1/\epsilon \rceil}$ and ν whether $\nu = \mathsf{SD}(C_0(\mathcal{U}), C_1(\mathcal{U}))$ or whether $|\nu - \mathsf{SD}(C_0(\mathcal{U}), C_1(\mathcal{U}))| > \epsilon$.

Consider now the real valued function $f_{\mathsf{SD}} : \{0,1\}^* \to \mathbb{R}$ defined as

$$f_{\mathsf{SD}}(C_0, C_1, 1^k) = \mathsf{SD}(C_0(\mathcal{U}), C_1(\mathcal{U}))$$

(note that the third parameter is ignored and is used only for padding purposes) and consider the class $\mathbb{R}$-TFAM as defined by Mahmoody and Xiao[MX10].

Definition 45 ($\mathbb{R}$-TFAM)**.** *A function $f : \{0,1\}^* \to \mathbb{R}$ is in $\mathbb{R}$-TFAM if for every $\epsilon \geq 1/\mathsf{poly}(n)$, the following relation $R = (R_{\mathsf{Yes}}, R_{\mathsf{No}})$ is in* AM*:*

1. $R_{\mathsf{Yes}} = \{(x, f(x)) \,|\, x \in \{0,1\}^*\}$
2. $R_{\mathsf{No}} = \left\{(x, y) \,\middle|\, x \in \{0,1\}^* \wedge \left|y - f(x)\right| \geq \epsilon\right\}$

Our protocol above implies, by definition, that $f_{\mathsf{SD}} \in \mathbb{R}$-TFAM. Furthermore, it holds that $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{BPP}^{\mathcal{O}_{f_{\mathsf{SD}}}}$, for any oracle $\mathcal{O}_{f_{\mathsf{SD}}}$ that on input $x \in \{0,1\}^n$ outputs a value y such that $\left|y - f_{\mathsf{SD}}(x)\right| \leq 1/n$. To see this, we notice that we can answer queries to the GapSD oracle of the form $(C_0, C_1, \nu, 1^\ell)$ as follows: First compute $y = \mathcal{O}_{f_{\mathsf{SD}}}(C_0, C_1, 1^{2\ell})$, then if $y < \nu + 1/(2\ell)$ return Yes, otherwise return No. This implies that $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{BPP}^{\mathbb{R}\text{-}\mathsf{TFAM}}$ by [MX10, Definition 3.2] (when choosing $\epsilon(n) = 1/n$). Finally, [MX10, Theorem 1.1] states that $\mathsf{BPP}^{\mathbb{R}\text{-}\mathsf{TFAM}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$, which as shown above implies that $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ as desired. □

We now state an important corollary of Theorem 44 which shows that there would be unlikely consequences if $\mathsf{UniqueSAT} \in \mathsf{BPP}^{\mathsf{GapSD}}$.

Corollary 46. *If* $\mathsf{UniqueSAT} \in \mathsf{BPP}^{\mathsf{GapSD}}$*, then* $\mathsf{NP} \subseteq \mathsf{AM} \cap \mathsf{coAM}$*.*

Proof. By definition it holds that $\mathsf{NP} \subseteq \mathsf{BPP}^{\mathsf{SAT}}$. Theorem 42 implies that $\mathsf{BPP}^{\mathsf{SAT}} \subseteq \mathsf{BPP}^{\mathsf{UniqueSAT}}$. If $\mathsf{UniqueSAT} \in \mathsf{BPP}^{\mathsf{GapSD}}$ then $\mathsf{BPP}^{\mathsf{UniqueSAT}} \subseteq \mathsf{BPP}^{\mathsf{GapSD}}$. Together with $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$ from Theorem 44, we get

$$\mathsf{NP} \subseteq \mathsf{BPP}^{\mathsf{SAT}} \subseteq \mathsf{BPP}^{\mathsf{UniqueSAT}} \subseteq \mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}\,,$$

and the corollary follows. □

4.2.2 Obfuscation

In this section, we define the statistically secure variant of approximately correct indistinguishability obfuscation (saiO) and its generalization that we call *statistically secure approximately correct correlation obfuscation* (sacO). We start with the generalized variant sacO first and then define saiO as a special case. The notion of correlation obfuscation, in contrast to standard indistinguishability obfuscation, does not require that the output of the obfuscator is *indistinguishable* for functionally equivalent circuits. Rather, it only requires that there is a noticeable correlation between the outputs.

Definition 47 (Approximately Correct Correlation Obfuscation)**.** *Let* O *be a* PPT *algorithm that takes the security parameter* 1^n *and a boolean circuit (with a single output bit) as inputs and produces a boolean circuit as output. For a circuit* C*, we let* $\mathsf{O}(1^n, C; r)$ *denote the output of running* O *on* C *with randomness* r*, and we let* $\mathsf{O}(1^n, C)$ *denote the distribution* $\mathsf{O}(1^n, C; r)$ *with uniform* r*. We say that* O *is a* $(1-\epsilon)$*-approximately correct and* $(1-\delta)$*-secure correlation obfuscator sacO if the following conditions hold:*

Approximate Correctness. *For any circuit* C *it holds that*

$$\Pr_{r,x}[\mathsf{O}(1^n, C; r)(x) = C(x)] \geq 1 - \epsilon(|C| \cdot n).$$

Correlation. *For any pair of circuits* C_1, C_2 *which compute the same function and such that* $|C_1| = |C_2|$ *it holds that* $\mathsf{SD}(\mathsf{O}(1^n, C_1), \mathsf{O}(1^n, C_2)) \leq \delta(|C_1| \cdot n)$*.*

The definition of statistically secure approximately correct indistinguishability obfuscation (saiO) follows by requiring negligible statistical distance δ.

Definition 48 (Approximately Correct Indistinguishability Obfuscation)**.** *Let* O *be a* $(1-\epsilon)$*-approximately correct and* $(1-\delta)$*-secure correlation obfuscator. We say that* O *is also a* $(1-\epsilon)$*-approximately correct statistically secure indistinguishability obfuscator (saiO) if there exists a negligible function* $\mathsf{negl}(|C| \cdot n)$ *such that for all circuits* C *it holds that* $\delta(|C| \cdot n) \leq \mathsf{negl}(|C| \cdot n)$*.*

4.2.3 Puncturable Pseudorandom Functions

We use a weak notion of puncturable pseudorandom function. This notion suffices for our results and follows trivially from the stronger standard definition.

Definition 49 (Puncturable Pseudorandom Functions)**.** *A pair of* PPT *algorithms* $(\mathsf{PRF}, \mathsf{Puncture})$ *is a puncturable pseudorandom function with one-bit output if, on input a key* $k \in \{0,1\}^n$ *or a punctured key* k^* *and an input value* $x \in \{0,1\}^n$*,* PRF *deterministically outputs a bit* b *and on input a key* $k \in \{0,1\}^n$ *and an input value* x_0*,* Puncture *outputs a punctured key* k^* *such that the following two properties are satisfied.*

Functionality Preserved Under Puncturing. *For all keys* k*, all input values* x_0*, all punctured keys* $k^* \leftarrow_\$ \mathsf{Puncture}(k, x_0)$*, and all input values* $x \neq x_0$*, it holds that*

$$\mathsf{PRF}(k^*, x) = \mathsf{PRF}(k, x).$$

Security *For every* PPT *adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *such that* $\mathcal{A}_1(1^n; r_1)$ *outputs an input value* x_0 *and state* st, *consider an experiment where* $\mathsf{k} \leftarrow\!\!\$ \{0,1\}^n$, $\mathsf{k}^* = \mathsf{Puncture}(\mathsf{k}, x_0; t)$, *and* $b \leftarrow\!\!\$ \{0,1\}$. *Then we have*

$$\left| \Pr_{k,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, \mathsf{PRF}(k, x_0); r_2) = 1] - \Pr_{k,b,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, b; r_2) = 1] \right| \leq \mathsf{negl}(n).$$

As observed by [BW13; BGI14; KPT+13] puncturable PRFs can, for example, be constructed from pseudorandom generators (and thereby one-way functions [HIL+99]) via the GGM tree-based construction [GGM84; GGM86].

4.3 Negative Results for sacO and saiO

We now prove our main theorem that sacO for a large class of parameters, in particular the saiO parameters, is impossible assuming one-way functions and $\mathsf{NP} \not\subseteq \mathsf{AM} \cap \mathsf{coAM}$.

Theorem 50 (Impossibility of sacO)**.** *If* $(1-\epsilon)$*-approximately correct,* $(1-\delta)$*-secure sacO for* P *exists, and there exists some polynomial* $\mathsf{poly}(|C|, n)$ *such that* $\delta(|C| \cdot n) \leq \frac{1}{3} - \frac{2}{3}\epsilon(|C| \cdot n) - \frac{1}{\mathsf{poly}(|C| \cdot n)}$*, then one-way functions do not exist or* $\mathsf{NP} \subseteq \mathsf{coAM} \cap \mathsf{AM}$.

By setting δ to be some negligible function, impossibility of saiO follows immediately as a corollary.

Corollary 51 (Impossibility of saiO)**.** *If* $(1-\epsilon)$*-approximately correct, saiO for* P *exists, and there exists some polynomial* $\mathsf{poly}(|C|, n)$ *such that* $\epsilon(|C| \cdot n) \leq \frac{1}{2} - \frac{1}{\mathsf{poly}(|C| \cdot n)}$*, then one-way functions do not exist or* $\mathsf{NP} \subseteq \mathsf{coAM} \cap \mathsf{AM}$.

Proof of Theorem 50. We define an efficiently sampleable distribution $X[\Psi]$ that is parametrized by a formula Ψ, and we define a reference distribution Y that should be parametrized by the size of Ψ and the number of variables in Ψ, but we omit the dependency on Ψ for readability. We note that in the introduction, we discussed to use $Y = X[\mathbf{0}]$, where $\mathbf{0}$ is a canonical representation of an unsatisfiable formula of the same size as Ψ. It is intuitive to think of Y as being indeed equal to $X[\mathbf{0}]$. However, for the sake of tightness, jumping ahead, we will use a slightly different distribution and note that this allows us to gain an additive term of δ in Claim 58.

As in the proof by Goldwasser and Rothblum [GR07; GR14] that we sketched in the introduction, we want to define $X[\Psi]$ (and Y) in a way such that the distributions are efficiently sampleable, that they are statistically close if Ψ is satisfiable, and that they are statistically far if Ψ is unsatisfiable, assuming one-way functions and sacO. If we manage to do so, then we suceed in showing that these assumptions imply the collapse of the polynomial hierarchy.

Our proof will rely on the promise problem (USAT, UNSAT) rather than the language SAT (See Section 4.2.1) and therefore, instead of using the gap statistical distance problem GapSD directly as Goldwasser-Rothblum, we will consider $\mathsf{BPP}^{\mathsf{GapSD}}$ to be able to accommodate the randomized reduction from SAT to USAT (See Theorem 42).

Our proof does not rely on complexity-theoretic techniques, except for proving the following claim and showing that the theorem follows from it.

Claim 52. *Assume that there is a formula-indexed distribution* $X[\Psi]$*, a reference distribution* Y*, a function* ν*, and a polynomial* $\mathsf{poly}(n)$ *such that the following three conditions are satisfied.*

(1) There is a uniform polynomial-time algorithm $\mathcal{A}$, that on input Ψ, constructs two polynomial-size randomized circuits that sample from $X[\Psi]$ and Y respectively.

(2) If Ψ is in UNSAT, *then $X[\Psi]$ is has statistical distance at most $\nu(n)$ from Y.*

(3) If Ψ is in USAT, *then $X[\Psi]$ has statistically distance at least $\nu(n) + \frac{1}{\mathsf{poly}(n)}$ from Y.*

Then USAT *is in* $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$.

Proof. Given that conditions (1), (2) and (3) are satisfied, we construct an algorithm $\mathcal{B}$ such that for all GapSD oracles and all formulae Ψ, $\mathcal{B}^{\mathsf{GapSD}}(\Psi)$ outputs 1 with probability 1 if $\Psi \in \mathsf{USAT}$ and 0 with probability 1 if $\Psi \in \mathsf{UNSAT}$. On input Ψ, the algorithm $\mathcal{B}$ runs $\mathcal{A}$ to get circuits for $X[\Psi]$ and Y and queries $(X[\Psi], Y, \nu(n), 1^{\mathsf{poly}(n)})$ to the GapSD oracle. $\mathcal{B}$ returns whatever the oracle returns. By properties (1), (2) and (3), the query that $\mathcal{B}$ makes is in $\mathsf{GapSD}_{\mathsf{Yes}}$ if $\Psi \in \mathsf{USAT}$ and in $\mathsf{GapSD}_{\mathsf{No}}$ if $\Psi \in \mathsf{UNSAT}$. Hence, $\mathcal{B}$ is correct and USAT is in $\mathsf{BPP}^{\mathsf{GapSD}}$. Moreover, due to Theorem 44, $\mathsf{BPP}^{\mathsf{GapSD}} \subseteq \mathsf{AM} \cap \mathsf{coAM}$. □

To obtain the main theorem, we need to show that USAT is in $\mathsf{BPP}^{\mathsf{GapSD}}$ implies that NP is in $\mathsf{AM} \cap \mathsf{coAM}$ which directly follows from Corollary 46 of Theorem 44. Thus, if we can show that a distributions as described in conditions (1), (2) and (3) exist, then the theorem follows. □

We now define $X[\Psi]$ and Y and then show that they satisfy (1), (2) and (3) assuming the existence of one-way functions and sacO with suitable correctness and security.

Definition 53 (Distribution)**.** *Let $\ell(n)$ be a sufficiently large polynomial designating the size to which all circuits are padded before being obfuscated. Let Ψ be a formula, let* (PRF, Puncture) *be a puncturable pseudorandom function, and let* O *be a $(1-\epsilon)$-correct, statistically $(1-\delta)$-secure approximate correlation obfuscator, where $\delta(|C| \cdot n) \leq \frac{1}{3} - \frac{2}{3}\epsilon(|C| \cdot n) - \frac{1}{\mathsf{poly}(|C| \cdot n)}$. We now define the distribution $X[\Psi]$ and Y, where the circuits $\mathsf{C}_X[k, b, s, \Psi]$ and $\mathsf{C}_{prf}[k]$ are defined to the right of the distributions.*

$X[\Psi](1^n)$	$\mathsf{C}_X[k,s,\Psi](x)$	$Y(1^n)$	$\mathsf{C}_{prf}[k](x)$
$k \leftarrow\!\!\$ \{0,1\}^n$	**if** $\Psi(x \oplus s) = 1$	$k \leftarrow\!\!\$ \{0,1\}^n$	**return** $\mathsf{PRF}(k,x)$
$s \leftarrow\!\!\$ \{0,1\}^n$	**return** $\mathsf{PRF}(k,x) \oplus 1$	$s \leftarrow\!\!\$ \{0,1\}^n$	
$C := \mathsf{C}_X[k,s,\Psi]$	**else**	$C := \mathsf{C}_{prf}[k]$	
$C' \leftarrow\!\!\$ \mathsf{O}(1^n, C)$	**return** $\mathsf{PRF}(k,x)$	$C' \leftarrow\!\!\$ \mathsf{O}(1^n, C)$	
return (k, s, C')		**return** (k, s, C')	

Claim 54 (Distribution)**.** *The distributions defined in Definition 53 satisfy the conditions demanded in Claim 52. I.e., there exists a function ν and a polynomial $\mathsf{poly}(n)$ such that they satisfy the following:*

(1) There is a uniform polynomial-time algorithm $\mathcal{A}$, that on input Ψ, constructs two polynomial-size randomized circuits that sample from $X[\Psi]$ and Y respectively.

(2) If Ψ is in UNSAT, *then $X[\Psi]$ is has statistical distance at most $\nu(n)$ from Y.*

(3) If Ψ is in USAT, *then $X[\Psi]$ has statistically distance at least $\nu(n) + \frac{1}{\mathsf{poly}(n)}$ from Y.*

We will first state two claims and a lemma that will allow us to prove Claim 54. We will then prove Claim 54 and afterwards prove the claims and the lemma.

Claim 55 (Efficient Sampling)**.** *There is a uniform polynomial-time algorithm $\mathcal{A}$, that on input Ψ, constructs two polynomial-size randomized circuits that sample from $X[\Psi]$ and Y respectively..*

Claim 56 (Statistical Proximity)**.** *For all formulae $\Psi \in \mathsf{UNSAT}$, $X[\Psi]$ has statistical distance at most $\delta(\ell(n)\cdot n)$ from Y.*

Lemma 57 (Statistical Distance)**.** *There exists a negligible function $\mathsf{negl}(n)$, such that for all formulae $\Psi \in \mathsf{USAT}$, $X[\Psi]$ has statistical distance at least $1-2\epsilon(\ell(n)\cdot n)-2\delta(\ell(n)\cdot n)-\mathsf{negl}(n)$ from Y.*

Proof of Claim 54. Condition (1) follows immediately from Claim 55. Condition (2) follows from Claim 56 for a function $\nu(n)=\delta(\ell(n)\cdot n)$. From Lemma 57, it follows that, if Ψ is in USAT, then $X[\Psi]$ has statistically distance at least $1-2\epsilon(\ell(n)\cdot n)-2\delta(\ell(n)\cdot n)-\mathsf{negl}(n)$ from Y. Combining this with the $\nu(n)$ obtained from Claim 56 we get that condition (3) holds, if there exists a polynomial $\mathsf{poly}(n)$, such that

$$\begin{aligned} & \delta(\ell(n)\cdot n)+\tfrac{1}{\mathsf{poly}(n)} \le 1-2\epsilon(\ell(n)\cdot n)-2\delta(\ell(n)\cdot n)-\mathsf{negl}(n) \\ \iff \quad & 3\delta(\ell(n)\cdot n) \le 1-2\epsilon(\ell(n)\cdot n)-\tfrac{1}{\mathsf{poly}(n)}-\mathsf{negl}(n) \\ \iff \quad & \delta(\ell(n)\cdot n) \le \frac{1}{3}-\frac{2}{3}\epsilon(\ell(n)\cdot n)-\tfrac{1}{\mathsf{poly}(n)}-\mathsf{negl}(n). \end{aligned} \tag{4.1}$$

And, since $\mathsf{negl}(n)$ is dominated by an inverse polynomial, Equation 4.1 is already ensured by Definition 53, condition (3) holds, and the claim follows. □

Proof of Claim 55. Sampling k and s is efficient and so is constructing $\mathsf{C}_X[k,s,\Psi]$ and $\mathsf{C}_{\mathsf{prf}}[k]$. Finally, from the efficiency of the obfuscator, it follows that $X[\Psi]$ and Y are efficiently sampleable by polynomial-size randomized circuits. □

Proof of Claim 56. For all unsatisfiable formulae Ψ, the circuits $\mathsf{C}_X[k,s,\Psi]$ and $\mathsf{C}_{\mathsf{prf}}[k]$ are functionally equivalent and of same size $\ell(n)$. Hence, by statistical security of the obfuscator, the distributions $(k,s,\mathsf{O}(1^n,\mathsf{C}_X[k,s,\Psi]))$ and $(k,s,\mathsf{O}(1^n,\mathsf{C}_{\mathsf{prf}}[k]))$ have statistical distance at most $\delta(\ell(n)\cdot n)$. □

We now turn to the most involved part of the proof, which is to show that Lemma 57 holds. In order to show that for all formulae $\Psi \in \mathsf{USAT}$, $X[\Psi]$ is statistically far from Y, we show that, if $\Psi \in \mathsf{USAT}$, then the distribution $X[\Psi]$ has a property that Y does not have. We state the property in two claims.

Claim 58. *For all x_0, it holds that*

$$\Pr_{(k,s,C')\leftarrow\$\, Y(1^n)}\left[C'(x_0\oplus s)\neq \mathsf{PRF}(k,x_0\oplus s)\right] \le \epsilon(\ell(n)\cdot n).$$

Claim 59. *If $\Psi \in \mathsf{USAT}$, then there exists x_Ψ, such that*

$$\Pr_{(k,s,C')\leftarrow\$\, X[\Psi](1^n)}\left[C'(x_\Psi\oplus s)\neq \mathsf{PRF}(k,x_\Psi\oplus s)\right] \ge 1-\epsilon(\ell(n)\cdot n)-2\delta(\ell(n)\cdot n)-\mathsf{negl}(n).$$

Proof of Lemma 57. Lemma 57 follows directly from Claim 58 and Claim 59, because the stated properties are statistical properties, i.e., we can give an inefficient distinguisher as follows: The distinguisher determines x_Ψ – which is exactly the satisfying assignment of Ψ – through exhaustive search and then, given a sample (k, s, C') from either $X[\Psi]$ or Y, checks whether $\mathsf{PRF}(k, \cdot)$ and C' differ on input $x_\Psi \oplus s$. If the sample is from $X[\Psi]$, they will differ with probability greater than $1 - \epsilon(\ell(n) \cdot n) - 2\delta(\ell(n), n) - \mathsf{negl}(n)$. If on the other hand the sample is from Y, then they will differ only with probability less than $\epsilon(\ell(n) \cdot n)$. This concludes the proof of Lemma 57, subject to proving the claims. □

It now remains to prove Claim 58 and Claim 59. The proof of the first property is relatively straightforward, while the proof of the second property contains the technical key arguments that we discussed above.

Proof of Claim 58. To prove the claim, we will argue that the following equalities hold:

$$\Pr_{(k,s,C') \leftarrow\$ Y(1^n)}\left[C'(x_0 \oplus s) \neq \mathsf{PRF}(k, x_0 \oplus s)\right] \tag{4.2}$$

$$= \Pr_{k,s \leftarrow\$ \{0,1\}^n, C' \leftarrow\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{prf}}[k])}\left[C'(x_0 \oplus s) \neq \mathsf{PRF}(k, x_0 \oplus s)\right] \tag{4.3}$$

$$= \Pr_{k,s \leftarrow\$ \{0,1\}^n, C' \leftarrow\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{prf}}[k])}\left[C'(s) \neq \mathsf{PRF}(k, s)\right] \tag{4.4}$$

$$\leq \quad \epsilon(\ell(n) \cdot n) \tag{4.5}$$

Equation 4.3 is simply a restatement of the claim. Given that s is uniformly and independently distributed, s and $x_0 \oplus s$ are distributed identically and therefore, also Equation 4.4 holds. Finally, Equation 4.4 simply checks whether an obfuscated circuit does not agree with the original circuit on a uniformly chosen input. This happens by definition of correctness with probability at most $\epsilon(\ell(n) \cdot n)$, yielding Equation 4.5 and concluding the proof. □

Proof of Claim 59. Let x_Ψ denote the accepting assignment of Ψ. We first define the game Game_1 as specified below and observe that

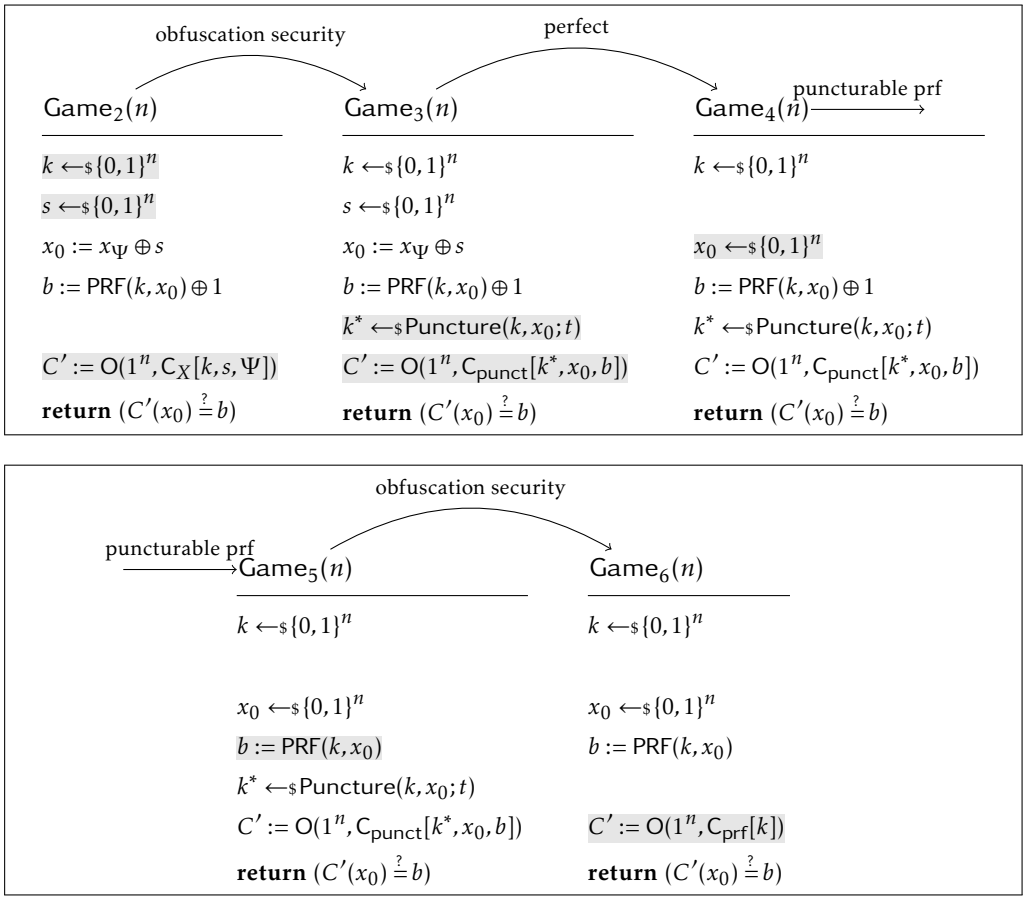
$$\Pr_{(k,s,C') \leftarrow\$ X[\Psi](1^n)}\left[C'(x_\Psi \oplus s) \neq \mathsf{PRF}(k, x_\Psi \oplus s)\right] = \Pr\left[\mathsf{Game}_1(n) = 1\right].$$

We will now bound this probability using a series of game hops. To specify the game hops, we need to specify an additional circuit $\mathsf{C}_{\mathsf{punct}}[k^*, x_0, b](x)$, that is parametrized by a punctured PRF key k^*, an input x_0 , and a bit b.

$\mathsf{Game}_1(n)$
$(k, s, C') \leftarrow\$ X[\Psi]$
$x_0 := x_\Psi \oplus s$
$b := \mathsf{PRF}(k, x_0) \oplus 1$
return $(C'(x_0) \stackrel{?}{=} b)$

$\mathsf{C}_{\mathsf{punct}}[k^*, x_0, b](x)$
if $x = x_0$
return b
else
return $\mathsf{PRF}(k^*, x)$

Note that Game_2 is a re-write of Game_1 by making $X[\Psi]$ explicit.



We will first bound the differences between each pair of consecutive games and then prove a bound for $\Pr[\mathsf{Game}_6(n) = 1]$.

Hop from Game_1 **to** Game_2**.** The changes between the two games are purely syntactic. I.e., the definition of the sampling process from $X[\Psi]$ is explicitely written down in Game_2. Therefore, the two games are perfectly equivalent, and it holds that

$$\Pr[\mathsf{Game}_1(n) = 1] = \Pr[\mathsf{Game}_2(n) = 1]. \tag{4.6}$$

Hop from Game_2 **to** Game_3**.** Here it is critical to observe that $\mathsf{C}_X[k, s, \Psi]$ and $\mathsf{C}_{\mathsf{punct}}[k^*, x_0, b]$ are functionally equivalent. Even though the key is punctured on $x_0 = x_\Psi \oplus s$ in $\mathsf{C}_{\mathsf{punct}}$, this makes no difference, since PRF is never invoked on x_0 in the circuit. Instead the circuit outputs the hardcoded value $b = \mathsf{PRF}(k, x_0) \oplus 1$ on input x_0, which is the same value output by $\mathsf{C}_X[k, s, \Psi]$. Therefore, the two circuits are functionally equivalent and it follows from the statistical security of the obfuscator that the statistical difference between the distributions of C' in the two games is at most $\delta(\ell(n) \cdot n)$. It follows, that also the distribution of the outputs of Game_2 and Game_3 have a statistical distance of at most $\delta(\ell(n) \cdot n)$. I.e.,

$$|\Pr[\mathsf{Game}_3(n) = 1] - \Pr[\mathsf{Game}_2(n) = 1]| \leq \delta(\ell(n) \cdot n). \tag{4.7}$$

Hop from Game_3 **to** Game_4**.** Since s is no longer known to the obfuscator in Game_3, $x_0 := x_\Psi \oplus s$ is simply a uniformly distributed value. Thus, x_0 is distributed identically in Game_3 and Game_4 and it follows that

$$\Pr[\mathsf{Game}_3(n) = 1] = \Pr[\mathsf{Game}_4(n) = 1]. \tag{4.8}$$

Hop from Game_4 **to** Game_5**.** Note that x_Ψ is no longer required to evaluate Game_4 and Game_5. Therefore, the two games can be evaluated efficiently. This allows us to bound the difference between the two games by the security of the puncturable pseudorandom function. To bound the difference between games $\mathsf{Game}_4(n)$ and $\mathsf{Game}_5(n)$, we construct a distinguisher $(\mathcal{A}_1, \mathcal{A}_2)$ with advantage

$$\tfrac{1}{2} \cdot |\Pr[\mathsf{Game}_4(n) = 1] - \Pr[\mathsf{Game}_5(n) = 1]|$$

against the puncturable PRF as follows:

$\mathcal{A}_1(1^n; r_1)$	$\mathcal{A}_2(\mathsf{st}, k^*, x_0, b; r_2)$
$x_0 \leftarrow\!\!\$ \{0,1\}^n$	$C' := \mathsf{O}(1^n, \mathsf{C}_{\mathsf{punct}}[k^*, x_0, b])$
return $(\bot, x_0)$	**return** $(C'(x_0) \stackrel{?}{=} b)$

Observe, that in the case where $\mathcal{A}_2$ receives the PRF value, it holds that

$$\Pr_{k,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, \mathsf{PRF}(k, x_0); r_2) = 1] = \Pr[\mathsf{Game}_5(n) = 1]. \tag{4.9}$$

If on the other hand, $\mathcal{A}_2$ receives a b chosen uniformly at random, then b is equal to $\mathsf{PRF}(k, x_0)$ and $\mathsf{PRF}(k, x_0) \oplus 1$ with probability $\frac{1}{2}$ respectively, and it holds that

$$\Pr_{k,b,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, b; r_2) = 1] = \frac{1}{2}\Pr[\mathsf{Game}_4(n) = 1] + \frac{1}{2}\Pr[\mathsf{Game}_5(n) = 1] \tag{4.10}$$

By security of the puncturable PRF, it must hold that

$$\left| \Pr_{k,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, \mathsf{PRF}(k, x_0); r_2) = 1] - \Pr_{k,b,r_1,t,r_2}[\mathcal{A}_2(\mathsf{st}, \mathsf{k}^*, x_0, b; r_2) = 1] \right| \leq \mathsf{negl}(n).$$

Combining this with Equation 4.9 and Equation 4.10 yields

$$\begin{aligned} & \left| \Pr[\mathsf{Game}_5(n) = 1] - \frac{1}{2}\Pr[\mathsf{Game}_4(n) = 1] - \frac{1}{2}\Pr[\mathsf{Game}_5(n) = 1] \right| \leq \mathsf{negl}(n) \\ \Longrightarrow \quad & \frac{1}{2}|\Pr[\mathsf{Game}_5(n) = 1] - \Pr[\mathsf{Game}_4(n) = 1]| \leq \mathsf{negl}(n) \\ \Longleftrightarrow \quad & |\Pr[\mathsf{Game}_5(n) = 1] - \Pr[\mathsf{Game}_4(n) = 1]| \leq 2\mathsf{negl}(n). \end{aligned} \tag{4.11}$$

Hop from Game_5 **to** Game_6**.** Here it is critical to observe that $\mathsf{C}_{\mathsf{punct}}[k^*, x_0, b]$ and $\mathsf{C}_{\mathsf{prf}}[k]$ are functionally equivalent. Even though the key is punctured on x_0 in $\mathsf{C}_{\mathsf{punct}}$, this makes no difference, since PRF is never invoked on x_0 in the circuit. Instead the circuit outputs the hardcoded value $b = \mathsf{PRF}(k, x_0)$ on input x_0. Therefore, the two circuits are functionally equivalent and it follows from the statistical security of the obfuscator that the statistical difference between

the distributions of C' in the two games is at most $\delta(\ell(n)\cdot n)$. It follows, that also the distribution of the outputs of Game_5 and Game_6 have a statistical distance of at most $\delta(\ell(n)\cdot n)$. I.e.,

$$|\Pr[\mathsf{Game}_5(n)=1] - \Pr[\mathsf{Game}_6(n)=1]| \leq \delta(\ell(n)\cdot n). \tag{4.12}$$

It remains to bound the probability $\Pr[\mathsf{Game}_6(n)=1]$. Observe, that x_0 is a uniformly chosen input unknown to the obfuscator. Further, the $\mathsf{Game}_6(n)$ simply checks whether the output of circuit C' is the correct output value of the obfuscated circuit. Therefore, the correctness of the obfuscator implies that

$$\Pr[\mathsf{Game}_6(n)=1] \geq 1-\epsilon(\ell(n)\cdot n). \tag{4.13}$$

Finally, combining Equation 4.13 with Equations 4.6, 4.7, 4.8, 4.11, and 4.12, we get

$$\begin{aligned}
&\Pr[\mathsf{Game}_1(n)=1]\\
\geq&\Pr[\mathsf{Game}_6(n)=1] - |\Pr[\mathsf{Game}_1(n)=1] - \Pr[\mathsf{Game}_6(n)=1]|\\
\geq&1-\epsilon(\ell(n)\cdot n) - 2\delta(\ell(n)\cdot n) - 2\mathsf{negl}(n)
\end{aligned}$$

thus concluding the proof of Claim 59 and Theorem 50. □

4.4 A positive result for Correlation Obfuscation

In this section, we instantiate approximately correct correlation obfuscation for a large class of weak parameters. The idea of the construction is fairly simple and is based on two observations. For circuits with only a single bit output, we can efficiently estimate the majority of the outputs by using random sampling. This estimation depends only on the function computed by a circuit and not on the circuit itself. Therefore an obfuscator that simply outputs the estimated majority is fully secure but only correct with probability about $\frac{1}{2}$. An obfuscator, that simply outputs the circuit itself, on the other hand, is not secure at all (statistical distance is 1), but is fully correct.

By combining these two obfuscators and outputting the majority with probability 2ϵ and the circuit itself with probability $1-2\epsilon$ we can construct a roughly $(1-\epsilon)$ approximatly correct and $(1-2\epsilon)$ secure obfuscator $\mathsf{O}_{\epsilon,\mu}$ as detailed below. The parameter μ is some inverse polynomial function that describes the amount of approximation error that we allow (and that affects the correctness of $\mathsf{O}_{\epsilon,\mu}$) when the obfuscators samples repeatedly from the output distribution of the circuit to see whether the circuit is closer to the constant 1 or constant 0 function.

For any circuit C, $\mathsf{in}(C)$ denotes the number of input wires. For $b\in\{0,1\}$, Const_b^i is a canonical circuit with input length i and constant output b. The Bernoulli distribution of a parameter $p\in[0,1]$ is defined by Ber_p, i.e., it holds that $\Pr_{b\leftarrow\$\mathrm{Ber}_p}[b=1]=p$ and $\Pr_{b\leftarrow\$\mathrm{Ber}_p}[b=0]=1-p$. Depending on the desired error parameter μ, the obfuscation proceeds as follows.

$\mathsf{O}_{\epsilon,\mu}(1^n, C)$

$b \leftarrow\!\!\$ \mathrm{Ber}_{2\epsilon}$
if $b = 1$:
  $m := \mathsf{EstMaj}(C, \mu, 1^n)$
  $C' := \mathsf{Const}_m^{\mathsf{in}(C)}$
else
  $C' := C$
return C'

$\mathsf{EstMaj}(C, \mu, 1^n)$

for $i := 1, \ldots, \left\lceil \frac{4n}{\mu^2} \right\rceil$
  $x_i \leftarrow\!\!\$ \{0,1\}^{\mathsf{in}(C)}$
  $y_i := C(x_i)$
return $\mathsf{maj}\left(y_1, \ldots, y_{\left\lceil \frac{4n}{\mu^2} \right\rceil}\right)$

Claim 60. *On input $(C, 1^n)$, the obfuscator $\mathsf{O}_{\epsilon,\mu}$ runs in time linear in $\frac{4n}{\mu^2}|C|$ plus the time needed to sample from $\mathrm{Ber}_{2\epsilon}$ and is an $(1-(\epsilon+\mu))$ approximately correct and $(1-2\epsilon)$ secure correlation obfuscator for circuits with single bit output.*

Proof. Efficiency follow by construction and so does security, because EstMaj only uses the input-output behaviour of the circuit which is the same for two functionally identical circuits. If the function induced by the circuit C is less than $\frac{\mu}{4}$ from being balanced (i.e., 1 with probability $\frac{1}{2}$ on a uniformly random input), then the correctness error is at most $\frac{\mu}{2}$, if $b = 1$, and 0, if $b = 0$ and hence, the overall correctness error is upper bounded by $(1-2\epsilon)\cdot 0 + 2\epsilon \cdot \frac{\mu}{2} = \epsilon\mu \leq \epsilon + \mu$. If the function induced by the circuit C outputs a fixed value, w.l.o.g. 1, with probability at least $\frac{1}{2} + \frac{\mu}{4}$, then via a Chernoff bound, the probability that $\mathsf{EstMaj}(C, \mu, 1^n)$ outputs 1 is at least $1 - \mathsf{negl}(n)$ and in that case, the correctness error is at most $\frac{1}{2} - \frac{\mu}{4}$ and else, the correctness error is at most 1. Hence, for the case that $b = 1$, we obtain an upper bound on the correctness error of $(\frac{1}{2} - \frac{\mu}{4})\cdot(1 - \mathsf{negl}(n)) + 1 \cdot \mathsf{negl}(n) = \frac{1}{2} - \frac{\mu}{4} + \mathsf{negl}(n)$. As before, when $b = 0$, the correctness error is 0 and hence, we obtain as upper bound on the correctness error $(1-2\epsilon)\cdot 0 + 2\epsilon\cdot(\frac{1}{2} - \frac{\mu}{4} + \mathsf{negl}(n)) = \epsilon - \frac{\mu}{2} \leq \epsilon + \mu$. □

4.5 Fom OWF to PKE using Correlation Obfuscation

By inspecting the Sahai-Waters [SW14] construction to transform a one-way function into a public-key encryption scheme (PKE) by using obfuscation, Bitansky and Vaikuntanathan [BV16] and Mahmoody et al. [MMN+16] observe that approximately correct iO suffices for this transformation. Both papers consider approximately correct variants of iO with "full" security, i.e., where the adversary has only negligible advantage in distinguishing obfuscations of two functionally equivalent circuits. As discussed in previous sections, approximately correct correlation obfuscation (sacO) with weaker security might still be useful. We therefore work out the exact correctness and security parameters required of a sacO for the Sahai-Waters transformation to work. Jumping ahead, we note that part of the bounds that we obtain here are ruled out by our impossibility result, but not all of them.

For much weaker parameters, we earlier gave a trivial construction of sacO. We do not deem this construction to be useful. As expected, there is a gap between the parameters that we can construct trivially and the parameters that we can rule out (else, we would have a proof that one-way functions imply the collapse of the polynomial hierarchy). Also, as expected, the trivial

bounds do not suffice to instantiate the Sahai-Waters construction (according to our analysis that we have reasons to believe is tight).

On the other hand, our impossibility result does not rule out all useful bounds for sacO. It is an interesting question to (1) show that also for the parameters in this small gap, sacO cannot exist, or (2) show a construction for these parameters, and/or (3) improve the parameters that are needed for meaningful applications. Note that even if it turns out that sacO for these parameters cannot exist, (3) could still be a fruitful research direction, because it might be helpful to weaken the parameters also on variants of acO with *computational* security in order to obtain constructions from weaker assumptions.

We will consider sacO with $(1-\delta)$-security and $(1-\epsilon)$ correctness, and we will also yield a PKE that does not achieve full correctness and that does not achieve full security. In some cases, as observed by Holenstein [Hol06], via amplification, it is possible to achieve full security and correctness with overwhelming probability. However, as we discuss now, amplification is not always possible.

4.5.1 Amplification

We define $(1-\epsilon_{\mathsf{PKE}})$-correct and $(\frac{1}{2}-\delta_{\mathsf{PKE}})$-secure PKE as follows.

Definition 61 (Approximate Public Key Encryption)**.** *Let* $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public key encryption scheme.*

Correctness. *We say that* PKE *is* $(1-\epsilon_{\mathsf{PKE}})$*-correct, if it holds that*

$$\Pr_{b,r_1,r_2}[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(b, pk; r_2)) = b, (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{PKE.KGen}(1^n; r_1)] \geq 1 - \epsilon_{\mathsf{PKE}}(n).$$

Security. *We say that* PKE *is* $(\frac{1}{2}-\delta_{\mathsf{PKE}})$*-secure, if for all efficient adversaries* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(n)$ *such that for a uniformly chosen bit* b

$$\Pr_{b,r_1,r_2,r_3}[\mathcal{A}(\mathsf{pk}, \mathsf{Enc}(b, \mathsf{pk}; r_2); r_3) = b, (\mathsf{pk}, \mathsf{sk}) \leftarrow_{\$} \mathsf{KGen}(1^n; r_1)] \leq \frac{1}{2} + \delta_{\mathsf{PKE}}(n) + \mathsf{negl}(n).$$

We would like to amplify such a scheme into "standard" PKE, where ϵ_{PKE} and δ_{PKE} are negligible. We now discuss via a counterexample why such an amplification is not generally possible. Take a bit encryption scheme that outputs the message bit with probability α and a random bit with probability $1-\alpha$ and where decryption is the identity function. This PKE scheme is $(\frac{1}{2}-\frac{\alpha}{2})$-secure and $(\frac{1}{2}+\frac{\alpha}{2})$-correct. Correctness parameters are thus only meaningful if ϵ_{PKE} and δ_{PKE} are bounded away from $\frac{1}{2}$ and if, moreover, there is a meaningful relationship between the security and the correctness parameter. Holenstein [Hol06] shows (and we use the presentation of Mahmoody et al. [MMN+16] here) that amplification is possible if there exists a polynomial $\mathsf{poly}(n)$ such that

$$(1 - 2\epsilon_{\mathsf{PKE}}(n))^2 > 2\delta_{\mathsf{PKE}}(n) + \frac{1}{\mathsf{poly}(n)}.$$

Note that Holenstein also shows a tightness result for his amplification technique with respect to restricted black-box reductions.

4.5.2 The Sahai-Waters Construction

We now present the Sahai-Waters [SW14] construction of a public-key encryption scheme from a one-way function. We recall that – by Håstad et al. [HIL+99], Goldreich, Goldwasser and Micali [GGM86], and several independent proofs [BW13; BGI14; KPT+13] that the GGM construction is a puncturable PRF – puncturable PRFs and OWFs are existentially equivalent. The key generation of the Sahai-Waters construction draws a key k for a puncturable PRF as the secret key sk and then outputs an obfuscation of the following circuit $\mathsf{C}_{\mathsf{SW}}[k]$ as a public key pk:

$\mathsf{C}_{\mathsf{SW}}[k](m,r)$

$r' := \mathsf{PRG}(r)$

$c := m \oplus \mathsf{PRF}(k, r')$

return (r', c)

The encryption algorithm $\mathsf{Enc}(\mathsf{pk}, m, r)$ interprets the public key pk as a circuit, runs it on (m,r) and returns the result as a ciphertext. Finally, for decryption of a pair (r', c), the decryption algorithm $\mathsf{Dec}(\mathsf{sk}, (r', c))$ outputs $m := c \oplus \mathsf{PRF}(\mathsf{sk}, r')$.

Claim 62 (Sahai-Waters)**.** *The Sahai-Waters construction instantiated with sacO with correctness $1-\epsilon$ and security $1-\delta$ yields a public-key encryption scheme with correctness error $\epsilon_{\mathsf{PKE}}(n) = \epsilon(|C| \cdot n)$ and a distinguishing advantage of $\delta_{\mathsf{PKE}}(n) = \delta(|C| \cdot n) + \epsilon(|C| \cdot n)$*

Before we prove this claim, we will first illustrate what this implies for the bounds on parameters allowing for Holenstein amplification. Combining the bound for Holenstein amplification with Claim 62, we get that

$$2\delta_{\mathsf{PKE}}(n) + \frac{1}{\mathsf{poly}(n)} < (1 - 2\epsilon_{\mathsf{PKE}}(n))^2 \tag{4.14}$$

$$\iff \quad 2\delta(|C| \cdot n) + 2\epsilon(|C| \cdot n) + \frac{1}{\mathsf{poly}(n)} < (1 - 2\epsilon(|C| \cdot n))^2 \tag{4.15}$$

$$\iff \quad \delta(|C| \cdot n) < \frac{1}{2} - 3\epsilon(|C| \cdot n) + 2\epsilon(|C| \cdot n)^2 - \frac{1}{2\mathsf{poly}(n)}. \tag{4.16}$$

We thus get the following corollary.

Corollary 63. *Any $(1-\epsilon)$ correct and $(1-\delta)$ secure sacO implies a construction of public key encryption from one-way functions, if there exists some polynomial $\mathsf{poly}(|C| \cdot n)$ such that*

$$\delta(|C| \cdot n) < \frac{1}{2} - 3\epsilon(|C| \cdot n) + 2\epsilon(|C| \cdot n)^2 - \frac{1}{\mathsf{poly}(|C| \cdot n)}.$$

Proof of Claim 62. Note that correctness of the encryption scheme is over a random message, the randomness of the key generation and the randomness of the encryption algorithm. The obfuscated circuit is therefore invoked on a uniformly random input and the probability that it does not output the correct ciphertext can thus be bounded by the correctness error of the obfuscator. Since the decryption of the scheme is perfectly correct, we thus get that $\epsilon_{\mathsf{PKE}}(n) = \epsilon(|C| \cdot n)$.

To prove securtity, we first define the following game

$\mathsf{Game}_1(n)$

$k \leftarrow\!\$ \{0,1\}^n$
$r \leftarrow\!\$ \{0,1\}^{n/2}$
$\mathsf{pk} \leftarrow\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k])$
$b \leftarrow\!\$ \{0,1\}$
$c := \mathsf{pk}(b, r)$
$b' \leftarrow\!\$ \mathcal{A}(\mathsf{pk}, c)$
return $(b' \stackrel{?}{=} b)$

and observe that for a uniformly chosen bit $b \leftarrow\!\$ \{0,1\}$

$$\Pr_{b,r_1,r_2,r_3}[\mathcal{A}(\mathsf{pk}, \mathsf{Enc}(b, \mathsf{pk}; r_2); r_3) = b,\ (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$ \mathsf{KGen}(1^n; r_1)] = \Pr[\mathsf{Game}_1(n) = 1].$$

We will now bound this probability using a series of game hops.

PRG security — obfuscation security — PRF security

$\mathsf{Game}_2(n)$	$\mathsf{Game}_3(n)$	$\mathsf{Game}_4(n)$	$\mathsf{Game}_5(n)$
$k \leftarrow\!\$ \{0,1\}^n$	$k \leftarrow\!\$ \{0,1\}^n$	$k \leftarrow\!\$ \{0,1\}^n$	$k \leftarrow\!\$ \{0,1\}^n$
$r \leftarrow\!\$ \{0,1\}^{n/2}$			
$r' := \mathsf{PRG}(r)$	$r' \leftarrow\!\$ \{0,1\}^n$	$r' \leftarrow\!\$ \{0,1\}^n$	$r' \leftarrow\!\$ \{0,1\}^n$
		$k^* \leftarrow\!\$ \mathsf{Puncture}(k, r'; t)$	$k^* \leftarrow\!\$ \mathsf{Puncture}(k, r'; t)$
$\mathsf{pk} \leftarrow\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k])$	$\mathsf{pk} \leftarrow\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k])$	$\mathsf{pk} \leftarrow\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k^*])$	$\mathsf{pk} \leftarrow\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k^*])$
$b \leftarrow\!\$ \{0,1\}$	$b \leftarrow\!\$ \{0,1\}$	$b \leftarrow\!\$ \{0,1\}$	$b \leftarrow\!\$ \{0,1\}$
			$s \leftarrow\!\$ \{0,1\}$
$c := (b \oplus \mathsf{PRF}(k, r'), r')$	$c := (b \oplus \mathsf{PRF}(k, r'), r')$	$c := (b \oplus \mathsf{PRF}(k, r'), r')$	$c := (b \oplus s, r')$
$b' \leftarrow\!\$ \mathcal{A}(\mathsf{pk}, c)$	$b' \leftarrow\!\$ \mathcal{A}(\mathsf{pk}, c)$	$b' \leftarrow\!\$ \mathcal{A}(\mathsf{pk}, c)$	$b' \leftarrow\!\$ \mathcal{A}(\mathsf{pk}, c)$
return $(b' \stackrel{?}{=} b)$	**return** $(b' \stackrel{?}{=} b)$	**return** $(b' \stackrel{?}{=} b)$	**return** $(b' \stackrel{?}{=} b)$

We will first bound the differences between each pair of consecutive games and then argue a bound for $\Pr[\mathsf{Game}_5(n) = 1]$.

Hop from Game_1 to Game_2. The change between the two games is that the ciphertext is now no longer computed using the obfuscated circuit. Instead, it is computed as specified in the unobfuscated circuit $\mathsf{C}_{\mathsf{SW}}[k]$. Since the input to the circuit is uniformly and independently distributed, we can bound the probability that the two computations differ by the correctness of the sacO. I.e., it holds that

$$|\Pr[\mathsf{Game}_1(n) = 1] - \Pr[\mathsf{Game}_2(n) = 1]| \leq \epsilon(|C| \cdot n). \tag{4.17}$$

Hop from Game_2 **to** Game_3**.** The change between the two games is that the bitstring r' is no longer the output of a PRG and instead a uniformly chosen random string. We can thus bound the difference between the two games using the security of the pseudorandom generator. I.e., we can construct a distinguisher $\mathcal{D}$ with advantage $|\Pr[\mathsf{Game}_2(n)=1]-\Pr[\mathsf{Game}_3(n)=1]|$ as follows

$\mathcal{D}(r';r_1)$

$k \leftarrow\!\!\$\, \{0,1\}^n$

$\mathsf{pk} \leftarrow\!\!\$\, \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k])$

$b \leftarrow\!\!\$\, \{0,1\}$

$c := (b \oplus \mathsf{PRF}(k, r'), r')$

$b' \leftarrow\!\!\$\, \mathcal{A}(\mathsf{pk}, c)$

return $(b' \stackrel{?}{=} b)$

Observe, that in the case where $\mathcal{D}$ receives the output of the PRG, it holds that

$$\Pr_{r,r_1}[\mathcal{D}(\mathsf{PRG}(r);r_1)=1] = \Pr[\mathsf{Game}_2(n)=1]. \tag{4.18}$$

If on the other hand, $\mathcal{D}$ receives an r' chosen uniformly at random, then it holds that

$$\Pr_{r',r_1}[\mathcal{D}(r';r_1)=1] = \Pr[\mathsf{Game}_3(n)=1]. \tag{4.19}$$

By definition of a secure PRG, there further exists a negligible function $\mathsf{negl}(n)$, such that

$$\left|\Pr_{r,r_1}[\mathcal{D}(\mathsf{PRG}(r);r_1)=1] - \Pr_{r',r_1}[\mathcal{D}(r';r_1)=1]\right| \leq \mathsf{negl}(n).$$

Combining this with Equation 4.18 and Equation 4.19, we get

$$|\Pr[\mathsf{Game}_2(n)=1]-\Pr[\mathsf{Game}_3(n)=1]| \leq \mathsf{negl}(n). \tag{4.20}$$

Hop from Game_3 **to** Game_4**.** In this hop, the obfuscated circuit is replaced. It is critical to observe, that if r' is *not in the range* of PRG, then the two circuits are functionally equivalent, since the PRF will never be invoked on the point the key is punctured on. In this case, the distance between the two games can therefore be bounded by the security of the sacO. If r' *is in the range* of PRG, then we have no guarantee, but this only occurs with probabilty $2^{-n/2}$. Thus it follows that

$$|\Pr[\mathsf{Game}_3(n)=1]-\Pr[\mathsf{Game}_4(n)=1]| \leq \delta(|C|\cdot n) + 2^{-n/2}. \tag{4.21}$$

Hop from Game_4 **to** Game_5**.** Note that in Game_5, the PRF value is replaced with a uniformly chosen random value. This allows us to bound the difference between the two games by the security of the puncturable pseudorandom function. To bound the difference between games Game_4 and Game_5, we construct a distinguisher $(\mathcal{D}_1, \mathcal{D}_2)$ with advantage

$$|\Pr[\mathsf{Game}_4(n)=1]-\Pr[\mathsf{Game}_5(n)=1]|$$

against the puncturable PRF as follows:

$\mathcal{D}_1(1^n; r_1)$	$\mathcal{D}_2(\mathsf{st}, k^*, r', s; r_2)$
$r' \leftarrow\!\!\$ \{0,1\}^n$	$\mathsf{pk} \leftarrow\!\!\$ \mathsf{O}(1^n, \mathsf{C}_{\mathsf{SW}}[k^*])$
return $(\perp, r')$	$b \leftarrow\!\!\$ \{0,1\}$
	$c := (b \oplus s)$
	$b' \leftarrow\!\!\$ \mathcal{A}(\mathsf{pk}, c)$
	return $(C'(x_0) \stackrel{?}{=} b)$

Observe, that in the case where $\mathcal{A}_2$ receives the PRF value, it holds that

$$\Pr_{k,r_1,t,r_2}[\mathcal{D}_2(\mathsf{st}, \mathsf{k}^*, r', \mathsf{PRF}(k, r'); r_2) = 1] = \Pr[\mathsf{Game}_4(n) = 1]. \tag{4.22}$$

If on the other hand, $\mathcal{D}_2$ receives an s chosen uniformly at random, it holds that

$$\Pr_{k,s,r_1,t,r_2}[\mathcal{D}_2(\mathsf{st}, \mathsf{k}^*, r', s; r_2) = 1] = \Pr[\mathsf{Game}_5(n) = 1] \tag{4.23}$$

By security of the puncturable PRF, it must hold that there exists a negligible function $\mathsf{negl}(n)$ such that

$$\left| \Pr_{k,r_1,t,r_2}[\mathcal{D}_2(\mathsf{st}, \mathsf{k}^*, r', \mathsf{PRF}(k, r'); r_2) = 1] - \Pr_{k,s,r_1,t,r_2}[\mathcal{D}_2(\mathsf{st}, \mathsf{k}^*, r', s; r_2) = 1] \right| \leq \mathsf{negl}(n)$$

Combining this with Equation 4.22 and Equation 4.23 yields

$$|\Pr[\mathsf{Game}_5(n) = 1] - \Pr[\mathsf{Game}_4(n) = 1]| \leq \mathsf{negl}(n) \tag{4.24}$$

It remains to bound the probability $\Pr[\mathsf{Game}_5(n) = 1]$. However, the ciphertext in Game_5 is simply a uniformly distributed random value that does not reveal any information about b. Therefore, it is easy to see that $\Pr[\mathsf{Game}_5(n) = 1] = \frac{1}{2}$. Combining this with Equations 4.17, 4.20, 4.21, and 4.24, we can conclude that

$$\Pr[\mathsf{Game}_1(n) = 1] \leq \frac{1}{2} + \delta(|C| \cdot n) + \epsilon(|C| \cdot n),$$

thus concluding the proof. □

Bibliography

[AGG+06] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. "On basing one-way functions on NP-hardness". In: *38th Annual ACM Symposium on Theory of Computing*. Ed. by Jon M. Kleinberg. Seattle, WA, USA: ACM Press, May 2006, pp. 701–710.

[AGG+10] Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. "Erratum for: on basing one-way functions on NP-hardness". In: *42nd Annual ACM Symposium on Theory of Computing*. Ed. by Leonard J. Schulman. Cambridge, MA, USA: ACM Press, June 2010, pp. 795–796.

[AMS+11] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, François-Xavier Standaert, and Christian Wachsmann. "A Formalization of the Security Features of Physical Functions". In: *2011 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 2011, pp. 397–412.

[BB04] Dan Boneh and Xavier Boyen. "Secure Identity Based Encryption Without Random Oracles". In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2004, pp. 443–459.

[BB15] Andrej Bogdanov and Christina Brzuska. "On Basing Size-Verifiable One-Way Functions on NP-Hardness". In: *TCC 2015: 12th Theory of Cryptography Conference, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 1–6.

[BBF16a] Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. *On Statistically Secure Obfuscation with Approximate Correctness*. Cryptology ePrint Archive, Report 2016/226. http://eprint.iacr.org/2016/226. 2016.

[BBF16b] Zvika Brakerski, Christina Brzuska, and Nils Fleischhacker. "On Statistically Secure Obfuscation with Approximate Correctness". In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 551–578.

[BCF+09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. "Foundations of Non-malleable Hash and One-Way Functions". In: *Advances in Cryptology – ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. Lecture Notes in Computer Science. Tokyo, Japan: Springer, Heidelberg, Germany, Dec. 2009, pp. 524–541.

[BCN+04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. "Universally Composable Protocols with Relaxed Set-Up Assumptions". In: *45th Annual Symposium on Foundations of Computer Science*. Rome, Italy: IEEE Computer Society Press, Oct. 2004, pp. 186–195.

[BCP+14] Nir Bitansky, Ran Canetti, Omer Paneth, and Alon Rosen. "On the existence of extractable one-way functions". In: *46th Annual ACM Symposium on Theory of Computing*. Ed. by David B. Shmoys. New York, NY, USA: ACM Press, May 2014, pp. 505–514.

[Ber08] Daniel J. Bernstein. "Proving Tight Security for Rabin-Williams Signatures". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Istanbul, Turkey: Springer, Heidelberg, Germany, Apr. 2008, pp. 70–87.

[BFM14] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. "Indistinguishability Obfuscation and UCEs: The Case of Computationally Unpredictable Sources". In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 188–205.

[BFS+11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. "Physically Uncloneable Functions in the Universal Composition Framework". In: *Advances in Cryptology – CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2011, pp. 51–70.

[BFW16] David Bernhard, Marc Fischlin, and Bogdan Warinschi. "On the Hardness of Proving CCA-Security of Signed ElGamal". In: *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9614. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, Heidelberg, Germany, Mar. 2016, pp. 47–69.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (Im)possibility of Obfuscating Programs". In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 1–18.

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. "On the (im)possibility of obfuscating programs". In: *Journal of the ACM* 59.2 (2012), p. 6.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. "Functional Signatures and Pseudorandom Functions". In: *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Hugo Krawczyk. Vol. 8383. Lecture Notes in Computer Science. Buenos Aires, Argentina: Springer, Heidelberg, Germany, Mar. 2014, pp. 501–519.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract)". In: *20th Annual ACM Symposium on Theory of Computing*. Chicago, IL, USA: ACM Press, May 1988, pp. 1–10.

[BJL+16] Christoph Bader, Tibor Jager, Yong Li, and Sven Schäge. "On the Impossibility of Tight Cryptographic Reductions". In: *Advances in Cryptology – EUROCRYPT 2016, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Vienna, Austria: Springer, Heidelberg, Germany, May 2016, pp. 273–304.

[BL13a] Foteini Baldimtsi and Anna Lysyanskaya. "On the Security of One-Witness Blind Signature Schemes". In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 82–99.

[BL13b] Andrej Bogdanov and Chin Ho Lee. "Limits of Provable Security for Homomorphic Encryption". In: *Advances in Cryptology – CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2013, pp. 111–128.

[BM08] Boaz Barak and Mohammad Mahmoody-Ghidary. *Merkle Puzzles are Optimal*. Cryptology ePrint Archive, Report 2008/032. `http://eprint.iacr.org/2008/032`. 2008.

[BM09] Boaz Barak and Mohammad Mahmoody-Ghidary. "Merkle Puzzles Are Optimal - An $O(n^2)$-Query Attack on Any Key Exchange from a Random Oracle". In: *Advances in Cryptology – CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2009, pp. 374–390.

[BM13] Boaz Barak and Mohammad Mahmoody. *Merkle's Key Agreement Protocol is Optimal: An $O(n^2)$ Attack on any Key Agreement from Random Oracles*. Manuscript. `http://www.cs.virginia.edu/~mohammad/files/papers/MerkleFull.pdf`. 2013.

[BM14] Christina Brzuska and Arno Mittelbach. "Indistinguishability Obfuscation versus Multi-bit Point Obfuscation with Auxiliary Input". In: *Advances in Cryptology – ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, Dec. 2014, pp. 142–161.

[BNP+03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. "The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme". In: *Journal of Cryptology* 16.3 (June 2003), pp. 185–215.

[BP13] Nir Bitansky and Omer Paneth. "On the impossibility of approximate obfuscation and applications to resettable cryptography". In: *45th Annual ACM Symposium on Theory of Computing*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. Palo Alto, CA, USA: ACM Press, June 2013, pp. 241–250.

[BR93] Mihir Bellare and Phillip Rogaway. "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols". In: *ACM CCS 93: 1st Conference on Computer and Communications Security*. Ed. by V. Ashby. Fairfax, Virginia, USA: ACM Press, Nov. 1993, pp. 62–73.

[BR96] Mihir Bellare and Phillip Rogaway. "The Exact Security of Digital Signatures: How to Sign with RSA and Rabin". In: *Advances in Cryptology – EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer, Heidelberg, Germany, May 1996, pp. 399–416.

[Bra79] Gilles Brassard. "Relativized cryptography". In: *20th Annual Symposium on Foundations of Computer Science*. San Juan, Puerto Rico: IEEE Computer Society Press, Oct. 1979, pp. 383–391.

[Bro05] Daniel R. L. Brown. *Breaking RSA May Be As Difficult As Factoring*. Cryptology ePrint Archive, Report 2005/380. http://eprint.iacr.org/2005/380. 2005.

[Bro16] Daniel R. L. Brown. "Breaking RSA May Be As Difficult As Factoring". In: *Journal of Cryptology* 29.1 (Jan. 2016), pp. 220–241.

[BT03] Andrej Bogdanov and Luca Trevisan. "On Worst-Case to Average-Case Reductions for NP Problems". In: *44th Annual Symposium on Foundations of Computer Science*. Cambridge, MA, USA: IEEE Computer Society Press, Oct. 2003, pp. 308–317.

[BT06] Andrej Bogdanov and Luca Trevisan. "On Worst-Case to Average-Case Reductions for NP Problems". In: *SIAM Journal on Computing* 36.4 (2006), pp. 1119–1159.

[BV16] Nir Bitansky and Vinod Vaikuntanathan. "Indistinguishability Obfuscation: From Approximate to Exact". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 67–95.

[BV98] Dan Boneh and Ramarathnam Venkatesan. "Breaking RSA May Not Be Equivalent to Factoring". In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 59–71.

[BW13] Dan Boneh and Brent Waters. "Constrained Pseudorandom Functions and Their Applications". In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 280–300.

[Can01] Ran Canetti. "Universally Composable Security: A New Paradigm for Cryptographic Protocols". In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 136–145.

[CDP+07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. "Universally Composable Security with Global Setup". In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 61–85.

[CF01] Ran Canetti and Marc Fischlin. "Universally Composable Commitments". In: *Advances in Cryptology – CRYPTO 2001*. Ed. by Joe Kilian. Vol. 2139. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2001, pp. 19–40.

[CHL02] Yan-Cheng Chang, Chun-Yun Hsiao, and Chi-Jen Lu. "On the Impossibilities of Basing One-Way Permutations on Central Cryptographic Primitives". In: *Advances in Cryptology – ASIACRYPT 2002*. Ed. by Yuliang Zheng. Vol. 2501. Lecture Notes in Computer Science. Queenstown, New Zealand: Springer, Heidelberg, Germany, Dec. 2002, pp. 110–124.

[CHZ14] Yu Chen, Qiong Huang, and Zongyang Zhang. "Sakai-Ohgishi-Kasahara Identity-Based Non-Interactive Key Exchange Revisited and More". In: *ACISP 14: 19th Australasian Conference on Information Security and Privacy*. Ed. by Willy Susilo and Yi Mu. Vol. 8544. Lecture Notes in Computer Science. Wollongong, NSW, Australia: Springer, Heidelberg, Germany, July 2014, pp. 274–289.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. "On the Limitations of Universally Composable Two-Party Computation Without Set-Up Assumptions". In: *Journal of Cryptology* 19.2 (Apr. 2006), pp. 135–167.

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. "On Obfuscation with Random Oracles". In: *TCC 2015: 12th Theory of Cryptography Conference, Part II*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9015. Lecture Notes in Computer Science. Warsaw, Poland: Springer, Heidelberg, Germany, Mar. 2015, pp. 456–467.

[CLO+02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. "Universally composable two-party and multi-party secure computation". In: *34th Annual ACM Symposium on Theory of Computing*. Montréal, Québec, Canada: ACM Press, May 2002, pp. 494–503.

[Cor00] Jean-Sébastien Coron. "On the Exact Security of Full Domain Hash". In: *Advances in Cryptology – CRYPTO 2000*. Ed. by Mihir Bellare. Vol. 1880. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2000, pp. 229–235.

[Cor02a] Jean-Sébastien Coron. "Optimal Security Proofs for PSS and Other Signature Schemes". In: *Advances in Cryptology – EUROCRYPT 2002*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Apr. 2002, pp. 272–287.

[Cor02b] Jean-Sébastien Coron. "Security Proof for Partial-Domain Hash Signature Schemes". In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2002, pp. 613–626.

[DFK+14] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. "Feasibility and Infeasibility of Secure Computation with Malicious PUFs". In: *Advances in Cryptology – CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2014, pp. 405–420.

[DFK+15] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. *Feasibility and Infeasibility of Secure Computation with Malicious PUFs*. Cryptology ePrint Archive, Report 2015/405. http://eprint.iacr.org/2015/405. 2015.

[DFK+16] Nico Döttling, Nils Fleischhacker, Johannes Krupp, and Dominique Schröder. "Two-Message, Oblivious Evaluation of Cryptographic Functionalities". In: *Advances in Cryptology – CRYPTO 2016, Part III*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9816. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 619–648.

[DH76] Whitfield Diffie and Martin E. Hellman. "Multiuser Cryptographic Techniques". In: *American Federation of Information Processing Societies: 1976 National Computer Conference*. Vol. 45. AFIPS Conference Proceedings. New York, NY, USA: AFIPS Press, June 1976, pp. 109–112.

[DHT12] Yevgeniy Dodis, Iftach Haitner, and Aris Tentes. "On the Instantiability of Hash-and-Sign RSA Signatures". In: *TCC 2012: 9th Theory of Cryptography Conference*. Ed. by Ronald Cramer. Vol. 7194. Lecture Notes in Computer Science. Taormina, Sicily, Italy: Springer, Heidelberg, Germany, Mar. 2012, pp. 112–132.

[DOP05] Yevgeniy Dodis, Roberto Oliveira, and Krzysztof Pietrzak. "On the Generic Insecurity of the Full Domain Hash". In: *Advances in Cryptology – CRYPTO 2005*. Ed. by Victor Shoup. Vol. 3621. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2005, pp. 449–466.

[DOR+08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data". In: *SIAM Journal on Computing* 38.1 (2008), pp. 97–139.

[DR03] Yevgeniy Dodis and Leonid Reyzin. "On the Power of Claw-Free Permutations". In: *SCN 02: 3rd International Conference on Security in Communication Networks*. Ed. by Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano. Vol. 2576. Lecture Notes in Computer Science. Amalfi, Italy: Springer, Heidelberg, Germany, Sept. 2003, pp. 55–73.

[DR12] Marten van Dijk and Ulrich Rührmair. *Physical Unclonable Functions in Cryptographic Protocols: Security Proofs and Impossibility Results*. Cryptology ePrint Archive, Report 2012/228. http://eprint.iacr.org/2012/228. 2012.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data". In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer, Heidelberg, Germany, May 2004, pp. 523–540.

[DS13] Ivan Damgård and Alessandra Scafuro. "Unconditionally Secure and Universally Composable Commitments from Physical Assumptions". In: *Advances in Cryptology – ASIACRYPT 2013, Part II*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8270. Lecture Notes in Computer Science. Bengalore, India: Springer, Heidelberg, Germany, Dec. 2013, pp. 100–119.

[FF13a] Marc Fischlin and Nils Fleischhacker. *Limitations of the Meta-Reduction Technique: The Case of Schnorr Signatures*. Cryptology ePrint Archive, Report 2013/140. http://eprint.iacr.org/2013/140. 2013.

[FF13b] Marc Fischlin and Nils Fleischhacker. "Limitations of the Meta-reduction Technique: The Case of Schnorr Signatures". In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Athens, Greece: Springer, Heidelberg, Germany, May 2013, pp. 444–460.

[FGK+11] Nils Fleischhacker, Felix Günther, Franziskus Kiefer, Mark Manulis, and Bertram Poettering. *Pseudorandom Signatures*. Cryptology ePrint Archive, Report 2011/673. http://eprint.iacr.org/2011/673. 2011.

[FGK+13] Nils Fleischhacker, Felix Günther, Franziskus Kiefer, Mark Manulis, and Bertram Poettering. "Pseudorandom signatures". In: *ASIACCS 13: 8th ACM Symposium on Information, Computer and Communications Security*. Ed. by Kefei Chen, Qi Xie, Weidong Qiu, Ninghui Li, and Wen-Guey Tzeng. Hangzhou, China: ACM Press, May 2013, pp. 107–118.

[Fis02] Marc Fischlin. "On the Impossibility of Constructing Non-interactive Statistically-Secret Protocols from Any Trapdoor One-Way Function". In: *Topics in Cryptology – CT-RSA 2002*. Ed. by Bart Preneel. Vol. 2271. Lecture Notes in Computer Science. San Jose, CA, USA: Springer, Heidelberg, Germany, Feb. 2002, pp. 79–95.

[FJS13] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. *On Tight Security Proofs for Schnorr Signatures*. Cryptology ePrint Archive, Report 2013/418. http://eprint.iacr.org/2013/418. 2013.

[FJS14] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. "On Tight Security Proofs for Schnorr Signatures". In: *Advances in Cryptology – ASIACRYPT 2014, Part I*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, Dec. 2014, pp. 512–531.

[FKM+15] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. *Efficient Unlinkable Sanitizable Signatures from Signatures with Re-Randomizable Keys*. Cryptology ePrint Archive, Report 2015/395. http://eprint.iacr.org/2015/395. 2015.

[FKM+16] Nils Fleischhacker, Johannes Krupp, Giulio Malavolta, Jonas Schneider, Dominique Schröder, and Mark Simkin. "Efficient Unlinkable Sanitizable Signatures from Signatures with Re-randomizable Keys". In: *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang. Vol. 9614. Lecture Notes in Computer Science. Taipei, Taiwan: Springer, Heidelberg, Germany, Mar. 2016, pp. 301–330.

[FLR+10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. "Random Oracles with(out) Programmability". In: *Advances in Cryptology – ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Singapore: Springer, Heidelberg, Germany, Dec. 2010, pp. 303–320.

[FMA12] Nils Fleischhacker, Mark Manulis, and Amir Azodi. *A Modular Framework for Multi-Factor Authentication and Key Exchange*. Cryptology ePrint Archive, Report 2012/181. http://eprint.iacr.org/2012/181. 2012.

[FMA14] Nils Fleischhacker, Mark Manulis, and Amir Azodi. "A Modular Framework for Multi-Factor Authentication and Key Exchange". In: *SSR 2014: 1st International Conference on Research on Security Standardisation*. Ed. by Liqun Chen and Chris Mitchell. Vol. 8893. Lecture Notes in Computer Science. London, UK: Springer, Heidelberg, Germany, Dec. 2014, pp. 190–214.

[FS10] Marc Fischlin and Dominique Schröder. "On the Impossibility of Three-Move Blind Signature Schemes". In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera: Springer, Heidelberg, Germany, May 2010, pp. 197–215.

[FS12] Dario Fiore and Dominique Schröder. "Uniqueness Is a Different Story: Impossibility of Verifiable Random Functions from Trapdoor Permutations". In: *TCC 2012: 9th Theory of Cryptography Conference*. Ed. by Ronald Cramer. Vol. 7194. Lecture Notes in Computer Science. Taormina, Sicily, Italy: Springer, Heidelberg, Germany, Mar. 2012, pp. 636–653.

[GBL08] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. "Improved Bounds on Security Reductions for Discrete Log Based Signatures". In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 93–107.

[GG98] Oded Goldreich and Shafi Goldwasser. *On the possibility of basing Cryptography on the assumption that $P \neq NP$*. Cryptology ePrint Archive, Report 1998/005. http://eprint.iacr.org/1998/005. 1998.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. "Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits". In: *54th Annual Symposium on Foundations of Computer Science*. Berkeley, CA, USA: IEEE Computer Society Press, Oct. 2013, pp. 40–49.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions (Extended Abstract)". In: *25th Annual Symposium on Foundations of Computer Science*. Singer Island, Florida: IEEE Computer Society Press, Oct. 1984, pp. 464–479.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions". In: *Journal of the ACM* 33.4 (Oct. 1986), pp. 792–807.

[GGS+13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. "Witness encryption and its applications". In: *45th Annual ACM Symposium on Theory of Computing*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. Palo Alto, CA, USA: ACM Press, June 2013, pp. 467–476.

[GIM+10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. "Interactive Locking, Zero-Knowledge PCPs, and Unconditional Cryptography". In: *Advances in Cryptology – CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2010, pp. 173–190.

[GKM+00] Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. "The Relationship between Public Key Encryption and Oblivious Transfer". In: *41st Annual Symposium on Foundations of Computer Science*. Redondo Beach, CA, USA: IEEE Computer Society Press, Nov. 2000, pp. 325–335.

[GMR01] Yael Gertner, Tal Malkin, and Omer Reingold. "On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates". In: *42nd Annual Symposium on Foundations of Computer Science*. Las Vegas, NV, USA: IEEE Computer Society Press, Oct. 2001, pp. 126–135.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. "A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks". In: *SIAM Journal on Computing* 17.2 (Apr. 1988), pp. 281–308.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *19th Annual ACM Symposium on Theory of Computing*. Ed. by Alfred Aho. New York City, NY, USA: ACM Press, May 1987, pp. 218–229.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[GR07] Shafi Goldwasser and Guy N. Rothblum. "On Best-Possible Obfuscation". In: *TCC 2007: 4th Theory of Cryptography Conference*. Ed. by Salil P. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 2007, pp. 194–213.

[GR14] Shafi Goldwasser and Guy N. Rothblum. "On Best-Possible Obfuscation". In: *Journal of Cryptology* 27.3 (July 2014), pp. 480–505.

[GW11] Craig Gentry and Daniel Wichs. "Separating succinct non-interactive arguments from all falsifiable assumptions". In: *43rd Annual ACM Symposium on Theory of Computing*. Ed. by Lance Fortnow and Salil P. Vadhan. San Jose, CA, USA: ACM Press, June 2011, pp. 99–108.

[HIL+99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A Pseudorandom Generator from any One-way Function". In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396.

[Hof11] Dennis Hofheinz. "Possibility and Impossibility Results for Selective Decommitments". In: *Journal of Cryptology* 24.3 (July 2011), pp. 470–516.

[Hol06] Thomas Holenstein. "Strengthening Key Agreement Using Hard-Core Sets". PhD thesis. ETH Zurich, 2006. ISBN: 3-86628-088-2.

[HR04] Chun-Yuan Hsiao and Leonid Reyzin. "Finding Collisions on a Public Road, or Do Secure Hash Functions Need Secret Coins?" In: *Advances in Cryptology – CRYPTO 2004*. Ed. by Matthew Franklin. Vol. 3152. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2004, pp. 92–105.

[HS07] Satoshi Hada and Kouichi Sakurai. "A Note on the (Im)possibility of Using Obfuscators to Transform Private-Key Encryption into Public-Key Encryption". In: *IWSEC 07: 2nd International Workshop on Security, Advances in Information and Computer Security*. Ed. by Atsuko Miyaji, Hiroaki Kikuchi, and Kai Rannenberg. Vol. 4752. Lecture Notes in Computer Science. Nara, Japan: Springer, Heidelberg, Germany, Oct. 2007, pp. 1–12.

[IL89] Russell Impagliazzo and Michael Luby. "One-way Functions are Essential for Complexity Based Cryptography (Extended Abstract)". In: *30th Annual Symposium on Foundations of Computer Science*. Research Triangle Park, North Carolina: IEEE Computer Society Press, Oct. 1989, pp. 230–235.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. "Founding Cryptography on Oblivious Transfer - Efficiently". In: *Advances in Cryptology – CRYPTO 2008*. Ed. by David Wagner. Vol. 5157. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2008, pp. 572–591.

[IR89] Russell Impagliazzo and Steven Rudich. "Limits on the Provable Consequences of One-Way Permutations". In: *21st Annual ACM Symposium on Theory of Computing*. Seattle, WA, USA: ACM Press, May 1989, pp. 44–61.

[IR90] Russell Impagliazzo and Steven Rudich. "Limits on the Provable Consequences of One-way Permutations". In: *Advances in Cryptology – CRYPTO'88*. Ed. by Shafi Goldwasser. Vol. 403. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1990, pp. 8–26.

[Kat07] Jonathan Katz. "Universally Composable Multi-party Computation Using Tamper-Proof Hardware". In: *Advances in Cryptology – EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. Lecture Notes in Computer Science. Barcelona, Spain: Springer, Heidelberg, Germany, May 2007, pp. 115–128.

[KK12] Saqib A. Kakvi and Eike Kiltz. "Optimal Security Proofs for Full Domain Hash, Revisited". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 2012, pp. 537–553.

[KKR+12] Stefan Katzenbeisser, Ünal Koçabas, Vladimir Rozic, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon". In: *Cryptographic Hardware and Embedded Systems – CHES 2012*. Ed. by Emmanuel Prouff and Patrick Schaumont. Vol. 7428. Lecture Notes in Computer Science. Leuven, Belgium: Springer, Heidelberg, Germany, Sept. 2012, pp. 283–301.

[KMN+14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. "One-Way Functions and (Im)Perfect Obfuscation". In: *55th Annual Symposium on Foundations of Computer Science*. Philadelphia, PA, USA: IEEE Computer Society Press, Oct. 2014, pp. 374–383.

[KMP16] Eike Kiltz, Daniel Masny, and Jiaxin Pan. "Optimal Security Proofs for Signatures from Identification Schemes". In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2016, pp. 33–61.

[KPT+13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. "Delegatable pseudorandom functions and applications". In: *ACM CCS 13: 20th Conference on Computer and Communications Security*. Ed. by Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung. Berlin, Germany: ACM Press, Nov. 2013, pp. 669–684.

[KSY11] Jonathan Katz, Dominique Schröder, and Arkady Yerukhimovich. "Impossibility of Blind Signatures from One-Way Permutations". In: *TCC 2011: 8th Theory of Cryptography Conference*. Ed. by Yuval Ishai. Vol. 6597. Lecture Notes in Computer Science. Providence, RI, USA: Springer, Heidelberg, Germany, Mar. 2011, pp. 615–629.

[LP09] Yehuda Lindell and Benny Pinkas. "A Proof of Security of Yao's Protocol for Two-Party Computation". In: *Journal of Cryptology* 22.2 (Apr. 2009), pp. 161–188.

[LPS+16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. "Output-Compressing Randomized Encodings and Applications". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 96–124.

[LV16] Tianren Liu and Vinod Vaikuntanathan. "On Basing Private Information Retrieval on NP-Hardness". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 372–386.

[Mau05] Ueli M. Maurer. "Abstract Models of Computation in Cryptography (Invited Paper)". In: *10th IMA International Conference on Cryptography and Coding*. Ed. by Nigel P. Smart. Vol. 3796. Lecture Notes in Computer Science. Cirencester, UK: Springer, Heidelberg, Germany, Dec. 2005, pp. 1–12.

[MMN+16] Mohammad Mahmoody, Ameer Mohammed, Soheil Nematihaji, Rafael Pass, and Abhi Shelat. "Lower Bounds on Assumptions Behind Indistinguishability Obfuscation". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 49–66.

[MMN16] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. "On the Impossibility of Virtual Black-Box Obfuscation in Idealized Models". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 18–48.

[MV10] Roel Maes and Ingrid Verbauwhede. "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions". In: ed. by Ahmad-Reza Sadeghi and David Naccache. Information Security and Cryptography. Springer, Heidelberg, Germany, 2010, pp. 3–37. ISBN: 978-3-642-14451-6.

[MX10] Mohammad Mahmoody and David Xiao. "On the Power of Randomized Reductions and the Checkability of SAT". In: *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*. IEEE Computer Society, 2010, pp. 64–75.

[Nie02] Jesper Buus Nielsen. "Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case". In: *Advances in Cryptology – CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 2002, pp. 111–126.

[NSW09] Gregory Neven, Nigel Smart, and Bogdan Warinschi. "Hash Function Requirements for Schnorr Signatures". In: *Journal of Mathematical Cryptology* 3.1 (2009), pp. 69–87.

[OSV+12] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. *Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions*. Cryptology ePrint Archive, Report 2012/143. http://eprint.iacr.org/2012/143. 2012.

[OSV+13] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. "Universally Composable Secure Computation with (Malicious) Physically Uncloneable Functions". In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Athens, Greece: Springer, Heidelberg, Germany, May 2013, pp. 702–718.

[Pap01] Ravikanth S. Pappu. "Physical One-Way Functions". PhD Thesis. Massachusetts Institute of Technology, 2001.

[Pas11] Rafael Pass. "Limits of provable security from standard assumptions". In: *43rd Annual ACM Symposium on Theory of Computing*. Ed. by Lance Fortnow and Salil P. Vadhan. San Jose, CA, USA: ACM Press, June 2011, pp. 109–118.

[PRT+02] Ravikanth S. Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. "Physical One-Way Functions". In: *Science* 297.5589 (Sept. 2002), pp. 2026–2030.

[PS16] Rafael Pass and Abhi Shelat. "Impossibility of VBB Obfuscation with Ideal Constant-Degree Graded Encodings". In: *TCC 2016-A: 13th Theory of Cryptography Conference, Part I*. Ed. by Eyal Kushilevitz and Tal Malkin. Vol. 9562. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Heidelberg, Germany, Jan. 2016, pp. 3–17.

[PS96] David Pointcheval and Jacques Stern. "Security Proofs for Signature Schemes". In: *Advances in Cryptology – EUROCRYPT'96*. Ed. by Ueli M. Maurer. Vol. 1070. Lecture Notes in Computer Science. Saragossa, Spain: Springer, Heidelberg, Germany, May 1996, pp. 387–398.

[PV05] Pascal Paillier and Damien Vergnaud. "Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log". In: *Advances in Cryptology – ASIACRYPT 2005*. Ed. by Bimal K. Roy. Vol. 3788. Lecture Notes in Computer Science. Chennai, India: Springer, Heidelberg, Germany, Dec. 2005, pp. 1–20.

[RD13] Ulrich Rührmair and Marten van Dijk. "PUFs in Security Protocols: Attack Models and Security Evaluations". In: *2013 IEEE Symposium on Security and Privacy*. Berkeley, CA, USA: IEEE Computer Society Press, May 2013, pp. 286–300.

[RKB10] Ulrich Rührmair, Stefan Katzenbeisser, and Heike Busch. "Strong PUFs: Models, Constructions, and Security Proofs". In: *Towards Hardware-Intrinsic Security*. Ed. by Ahmad-Reza Sadeghi and David Naccache. Springer, Heidelberg, Germany, 2010, pp. 79–96.

[RLB+08] Andy Rupp, Gregor Leander, Endre Bangerter, Alexander W. Dent, and Ahmad-Reza Sadeghi. "Sufficient Conditions for Intractability over Black-Box Groups: Generic Lower Bounds for Generalized DL and DH Problems". In: *Advances in Cryptology – ASIACRYPT 2008*. Ed. by Josef Pieprzyk. Vol. 5350. Lecture Notes in Computer Science. Melbourne, Australia: Springer, Heidelberg, Germany, Dec. 2008, pp. 489–505.

[RSS09] Ulrich Rührmair, Jan Sölter, and Frank Sehnke. *On the Foundations of Physical Unclonable Functions*. Cryptology ePrint Archive, Report 2009/277. `http://eprint.iacr.org/2009/277`. 2009.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. "Notions of Reducibility between Cryptographic Primitives". In: *TCC 2004: 1st Theory of Cryptography Conference*. Ed. by Moni Naor. Vol. 2951. Lecture Notes in Computer Science. Cambridge, MA, USA: Springer, Heidelberg, Germany, Feb. 2004, pp. 1–20.

[Rud92] Steven Rudich. "The Use of Interaction in Public Cryptosystems (Extended Abstract)". In: *Advances in Cryptology – CRYPTO'91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1992, pp. 242–251.

[Rüh10] Ulrich Rührmair. "Oblivious Transfer Based on Physical Uncloneable Functions". In: *TRUST 2010: 3rd International Conference on Trust and Trustworthy Computing*. Ed. by Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi. Vol. 6101. Lecture Notes in Computer Science. Berlin, Germany: Springer, Heidelberg, Germany, June 2010, pp. 430–440.

[Sch11] Sven Schäge. "Tight Proofs for Signature Schemes without Random Oracles". In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Tallinn, Estonia: Springer, Heidelberg, Germany, May 2011, pp. 189–206.

[Sch90] Claus-Peter Schnorr. "Efficient Identification and Signatures for Smart Cards". In: *Advances in Cryptology – CRYPTO'89*. Ed. by Gilles Brassard. Vol. 435. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1990, pp. 239–252.

[Sch91] Claus-Peter Schnorr. "Efficient Signature Generation by Smart Cards". In: *Journal of Cryptology* 4.3 (1991), pp. 161–174.

[Seu12] Yannick Seurin. "On the Exact Security of Schnorr-Type Signatures in the Random Oracle Model". In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer, Heidelberg, Germany, Apr. 2012, pp. 554–571.

[SFM+11] Immanuel Schweizer, Nils Fleischhacker, Max Mühlhäuser, and Thorsten Strufe. "SDF – Solar-Aware Distributed Flow in Wireless Sensor Networks". In: *LCN 2011: 36th IEEE Conference on Local Computer Networks*. Ed. by Tom Pfeifer, Anura Jayasumana, and Nils Aschenbruck. Bonn, Germany: IEEE, Oct. 2011, pp. 382–390.

[SFS+16] Jonas Schneider, Nils Fleischhacker, Dominique Schröder, and Michael Backes. "Efficient Cryptographic Password Hardening Services From Partially Oblivious Commitments". In: *ACM CCS 16: 23rd Conference on Computer and Communications Security*. Ed. by Christopher Kruegel, Andrew Myers, and Shai Halevi. Vienna, Austria: ACM Press, Oct. 2016, pp. 1192–1203.

[Sha49] Claude E. Shannon. "Communication theory of secrecy systems". In: *Bell Systems Technical Journal* 28.4 (1949), pp. 656–715.

[Sho97] Victor Shoup. "Lower Bounds for Discrete Logarithms and Related Problems". In: *Advances in Cryptology – EUROCRYPT'97*. Ed. by Walter Fumy. Vol. 1233. Lecture Notes in Computer Science. Konstanz, Germany: Springer, Heidelberg, Germany, May 1997, pp. 256–266.

[Sim98] Daniel R. Simon. "Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?" In: *Advances in Cryptology – EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Espoo, Finland: Springer, Heidelberg, Germany, May 1998, pp. 334–345.

[SV97] Amit Sahai and Salil P. Vadhan. "A Complete Promise Problem for Statistical Zero-Knowledge". In: *38th Annual Symposium on Foundations of Computer Science*. Miami Beach, Florida: IEEE Computer Society Press, Oct. 1997, pp. 448–457.

[SW14] Amit Sahai and Brent Waters. "How to use indistinguishability obfuscation: deniable encryption, and more". In: *46th Annual ACM Symposium on Theory of Computing*. Ed. by David B. Shmoys. New York, NY, USA: ACM Press, May 2014, pp. 475–484.

[Val84] Leslie G. Valiant. "A Theory of the Learnable". In: *Communications of the Association for Computing Machinery* 27.11 (Nov. 1984), pp. 1134–1142.

[VV85] Leslie G. Valiant and Vijay V. Vazirani. "NP is as Easy as Detecting Unique Solutions". In: *17th Annual ACM Symposium on Theory of Computing*. Ed. by Robert Sedgewick. Providence, RI, USA: ACM Press, May 1985, pp. 458–463.

[Wat05] Brent R. Waters. "Efficient Identity-Based Encryption Without Random Oracles". In: *Advances in Cryptology – EUROCRYPT 2005*. Ed. by Ronald Cramer. Vol. 3494. Lecture Notes in Computer Science. Aarhus, Denmark: Springer, Heidelberg, Germany, May 2005, pp. 114–127.

[ZCC+15] Zongyang Zhang, Yu Chen, Sherman S. M. Chow, Goichiro Hanaoka, Zhenfu Cao, and Yunlei Zhao. "Black-Box Separations of Hash-and-Sign Signatures in the Non-Programmable Random Oracle Model". In: *ProvSec 2015: 9th International Conference on Provable Security*. Ed. by Man Ho Au and Atsuko Miyaji. Vol. 9451. Lecture Notes in Computer Science. Kanazawa, Japan: Springer, Heidelberg, Germany, Nov. 2015, pp. 435–454.

[ZZC+14] Jiang Zhang, Zhenfeng Zhang, Yu Chen, Yanfei Guo, and Zongyang Zhang. "Black-Box Separations for One-More (Static) CDH and Its Generalization". In: *Advances in Cryptology – ASIACRYPT 2014, Part II*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8874. Lecture Notes in Computer Science. Kaoshiung, Taiwan, R.O.C.: Springer, Heidelberg, Germany, Dec. 2014, pp. 366–385.