

ON THE COMPLEXITY OF BLIND SIGNATURES

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doktor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Inform. Dominique Schröder

geboren in Wolfsburg.



Referenten: Dr. Marc Fischlin
Prof. Dr. Jonathan Katz

Tag der Einreichung: 10. Oktober 2010
Tag der mündlichen Prüfung: 18. November 2010

Darmstadt, 2010
Hochschulkennziffer: D 17

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit – abgesehen von den in ihr ausdrücklich genannten Hilfen – selbständig verfasst habe.

Wissenschaftlicher Werdegang

Oktober 2002 - September 2006. Studium der Informatik an der Technische Universität Braunschweig mit Nebenfach Mathematik.

seit Oktober 2006. Wissenschaftlicher Mitarbeiter in der Emmy-Noether-Forschungsgruppe “MiniCrypt” unter der Leitung von Dr. Marc Fischlin der Technischen Universität Darmstadt.

Juli 2010 - Oktober 2010. Forschungsaufenthalt an der University of Maryland, USA unter der Leitung von Prof. Dr. Jonathan Katz.

Journal Publications

1. M. Fischlin und D. Schröder. “Security of Blind Signatures Under Aborts und Applications to Adaptive Oblivious Transfer” *To appear in the Journal of Mathematical Cryptology*.

Peer Reviewed Conference und Workshop Publications

1. M. Fischlin und D. Schröder. “On the Impossibility of Three-Move Blind Signature Schemes” *Advances in Cryptology - EUROCRYPT 2010*. Lecture Notes in Computer Science, Volume 6110, pp. 197-215, Springer-Verlag, 2010.
2. O.Eikemeier, M. Fischlin, J.-F. Goetzmann, A. Lehmann, D. Schröder, P. Schröder, und D. Wagner. “History-Free Aggregate Message Authentication Codes” *Conference on Security and Cryptography for Networks — SCN’10*. Lecture Notes in Computer Science, Volume 6280, pp. 462-479, Springer-Verlag, 2010.
3. Alexander W. Dent, M. Fischlin, M. Manulis, D. Schröder, und M. Stam. “Confidential Signatures und Deterministic Signcryption” *Public-Key Cryptography — PKC 2010*. Lecture Notes in Computer Science, Volume 6056, pp. 462-479, Springer-Verlag, 2010.
4. C. Brzuska, M. Fischlin, A. Lehmann und D. Schröder. “Unlinkability of Sanitizable Signatures” *Public-Key Cryptography — PKC 2010*. Lecture Notes in Computer Science, Volume 6280, pp. 309-328, Springer-Verlag, 2010.
5. M. Rückert, M. Schneider und D. Schröder. “Efficient Generic Constructions for Verifiably Encrypted Signatures Without NIZKs” *Applied Cryptography und Network Security — ACNS ’10*. Lecture Notes in Computer Science, Volume 6280, pp. 309-328, Springer-Verlag, 2010.
6. C. Brzuska, H. Busch, Ö. Dagdelen, M. Fischlin, M. Franz, S. Katzenbeisser, M. Manulis, C. Onete, A. Peter, B. Poettering und D. Schröder. “Redactable Signatures for Tree-Structured Data: Definitions und Constructions” *Applied Cryptography und Network Security — ACNS ’10*. Lecture Notes in Computer Science, Volume 6280, pp. 309-328, Springer-Verlag, 2010.
7. M. Rückert und D. Schröder. “Fair Partially Blind Signature” *The 3rd International Conference on Cryptology in Africa — AFRICACRYPT ’10*. Lecture Notes in Computer Science, Volume 6280, pp. 309-328, Springer-Verlag, 2010.

8. D. Galindo, B. Libert, M. Fischlin, G. Fuchsbauer, A. Lehmann, M. Manulis und D. Schröder. “Public-Key Encryption with Non-interactive Opening: New Constructions und Stronger Definitions” *The 3rd International Conference on Cryptology in Africa — AFRICACRYPT '10*. Lecture Notes in Computer Science, Volume 6280, pp. 309-328, Springer-Verlag, 2010.
9. M. Fischlin und D. Schröder. “Security of Blind Signatures Under Aborts” *Public-Key Cryptography — PKC 2009*. Lecture Notes in Computer Science, Volume 5443, pp. 297-316, Springer-Verlag, 2009.
10. C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, und F. Volk. “Security of Sanitizable Signatures Revisited” *Public-Key Cryptography — PKC 2009*. Lecture Notes in Computer Science, Volume 5443, pp. 317-336, Springer-Verlag, 2009.
11. C. Brzuska, F. Fischlin, A. Lehmann und D. Schröder. “Sanitizable Signatures: How to Partially Delegate Control for Authenticated Data” *Biosig 2009*. Lecture Notes in Informatics (LNI), Volume 155, pp. 117-128, 2009.
12. M. Rückert und D. Schröder. “Security of Verifiably Encrypted Signatures and a Construction Without Rndom Oracles” *Pairings 2009*. Lecture Notes in Computer Science, Volume 5671, pp. 17-34, Springer-Verlag, 2009.
13. M. Rückert und D. Schröder. “Aggregate und Verifiably Encrypted Signatures from Multilinear Maps without Rndom Oracles” *ISA 2009*. Lecture Notes in Computer Science, Volume 5576, pp. 750-759, Springer-Verlag, 2009.

Acknowledgments

First and foremost I would like to thank Marc Fischlin, under whose supervision I have had the amazing opportunity to do my Ph.D. At the beginning Marc worked very close with me and helped me find my own research directions, always keeping the bigger picture in mind. Throughout our collaboration, he always took the time to discuss my ideas and guided me with his great experience. Marc taught me to write papers and research grants; he helped me with my talks and provided me with many opportunities to work with researchers. Thanks to Marc I have had the opportunity to spend one month in New York in December 2009, and three months in Maryland in the summer 2010. I am truly grateful for his support and help; and I am lucky to have had Marc as my advisor. Aside from the scientific aspects of our work, Marc was always very friendly and helpful. I believe that our relationship is much closer than a regular student/advisor relationship and his (non-academic) advise was also always very important for me.

I am also deeply grateful to Jonathan Katz. During the summer of 2010 I have had the pleasure and the honor to visit him in Maryland. Jonathan is not only a fascinating researcher who is always happy to work on interesting problems, but he is also a very friendly, straight forward, and warm person. I am very pleased that Jonathan agreed to be the co-referee of this thesis and I am very thankful that he traveled to Darmstadt to attend my defense. I am really looking forward to do my postdoc under his guidance.

I would also thank Jonathan's research group for the nice summer. In particular Dov Gordan for helping me to find an apartment and Arkady Yerukhimovich for the nice collaboration, all discussions, and for proofreading parts of my thesis.

I owe Alexander W. Dent, Dominique Unruh, and Ivan Visconti a big thank you for their support and their helpful suggestions. I am looking forward to working with you again.

I would like to additionally credit Johannes Buchmann, Matthias Hollick, and Thorsten Strufe for joining my committee.

During my Ph.D. studies, I have had the pleasure to work with many great researchers on interesting projects. I want to thank all my co-authors for the fruitful discussions, all comments that improved my research, and more importantly, for the great working atmosphere.

I want to thank all my office mates for the great years we spent in Darmstadt. In particular Vangelis Karatsiolis for the nice and warm welcome at the beginning and Markus Rückert, with whom I shared an office the last years. Markus and I spent many hours discussing our work on the white board and I owe him a big thank you for motivating me to stay in Darmstadt.

I also enjoyed working with the whole minicrypt group. I would therefore like to thank Anja, Christina, Cristina, Özi, and Paul for the nice time in Darmstadt. Furthermore, thanks to the entire CDC crypto group for the fruitful and friendly collaboration and for their technical and moral support. In particular, many thanks to Michael “the greenhorn” Schneider for all Fridays, and Marita Skrobic for all her help in non-scientific matters and for all the nice conversations. Thanks a lot for your support.

René Iser helped me proofreading my thesis. Thank you very much.

I would also like to thank CASED, ECRYPT II, the German Academic Exchange Service - DAAD, the German Research Foundation - DFG, and ECRYPT financing my Ph.D. over the last years.

The great support of my parents during the last few years played also an important role. I could not have reached this level, however, without my parents constant support. They always took the time to listen as I told them about my achievements and frustration, and they understood my feelings most of the time. Both Ulrike and Barthel were always interested about my life in Darmstadt and they also provided a great place to relax in Cologne. Ulrike and Barthel’s support and their suggestions were (and will always be) very important for me, and I really appreciate everything they did to help me.

Finally, I want to thank my lovely wife Heike. Heike and I have had the pleasure to share a second office in Darmstadt. It is somehow scary that we both do research in (more or less) the same area. The good aspect about this is that Heike could proofread parts of this thesis and that she fully understood my enthusiasm for my work. But, even more importantly, she reminded me that there are many more important things than this thesis, and thanks to her we always managed to have also “crypto-free” life. I am very lucky to have her on my side and I do not understand how she managed to be so patient with me. I truly believe that Heike’s endless support helped me to finish my Ph.D. and to handle all frustration. I am deeply grateful for every minute and I will endeavor to return at least a part of this favor.

Abstract

Blind signature schemes provide the functionality of a carbon copy envelope: The user (receiver) puts his message into this envelope and hands it over to the signer (sender). The signer in return signs the envelope and gives it back to the user who recovers the original signed message out of the envelope. Security says that the signer remains oblivious about the message (blindness), but at the same time the receiver cannot output any additional message/signature pair (unforgeability). Classical applications of blind signatures include e-cash and e-voting.

Blind signature schemes are an important cryptographic primitive and many constructions have been proposed in the literature. These instantiations differ mainly in round complexity, underlying computational assumptions, and the model in which the proof of security is given. However, the *minimal requirements* for blind signatures in terms of round complexity and computational assumptions without assuming setup assumptions are unknown. This thesis addresses both of these questions.

For the study of the round complexity, this thesis investigates the possibility of proving the security of a more general class of three-move blind signature schemes. We show that finding security proofs for these schemes via black-box reductions in the standard model is hard. Characteristic for this class is that it is publicly decidable from the transcript if the user can derive a valid signature, or not.

Regarding the computational assumptions, this thesis first shows that the class of unique blind signature schemes can be used to build oblivious transfer protocols in a black-box way. These blind signature schemes have at most one signature per message and public key. It is well known that oblivious transfer cannot be constructed from one-way functions in a black-box fashion. Thus, this result also holds for (regular) blind signature schemes.

Moreover, this thesis rules out black-box constructions of blind signature schemes from one-way functions. In fact, this thesis rules out constructions from a random permutation oracle. This separation holds even for schemes signing 1-bit messages that achieve security only against honest-but-curious behavior.

Zusammenfassung

Blinde Signaturen ähneln einem Briefumschlag, der aus Kohlepapier besteht. Der Signierer (Sender) unterschreibt dabei auf diesem Kohlepapier, ohne das im Briefumschlag liegende Dokument zu sehen. Danach erhält der Empfänger den Briefumschlag und somit die unterschriebene Nachricht. Interactive Signaturverfahren zwischen einem Sender und einem Empfänger nennt man blinde Signaturverfahren, wenn der Sender (Signierer) nicht "sieht", welche Nachricht er unterschreibt (Blindheit). Gleichzeitig darf der Empfänger nicht in der Lage sein, mehr unterschriebene Nachrichten auszugeben, als Protokollausführungen stattfanden (Unfälschbarkeit). Typische Anwendungen dieser Signaturverfahren sind unter anderem *e-cash* oder *e-voting*.

Blinde Signaturen stellen ein wichtiges Primitiv in der Kryptographie dar. Obwohl diese Signaturverfahren seit vielen Jahren breit erforscht wurden, beruhen die bekannten Lösungen entweder auf sehr starken Annahmen, oder weisen einen hohen Interaktionsaufwand auf. Aus diesem Grund befasst sich diese Dissertation sowohl mit dem Interaktionssaufwand, als auch mit den minimalen Annahmen von blinden Signaturen.

Diese Arbeit beweist, dass es für eine große Klasse von blinden Signaturverfahren, bei denen höchstens drei Interaktionen zwischen dem Sender und dem Empfänger stattfinden, keinen *Black-Box* Beweis im Standardmodell gibt. Charakteristisch für diese Klasse ist, dass man von der öffentlichen Kommunikation entscheiden kann, ob der Empfänger eine gültige Signatur erhält, oder nicht.

Des Weiteren wird gezeigt, dass die Gruppe von eindeutigen blinden Signaturverfahren in *Oblivious-Transfer*-Protokolle überführt werden kann. Diese Gruppe zeichnet sich dadurch aus, dass jede Nachricht pro öffentlichem Schlüssel genau eine Signatur besitzt. Von *Oblivious-Transfer*-Verfahren ist bereits bekannt, dass diese nicht aus *one-way* Funktionen konstruiert (*black-box*) werden kann. Somit gilt dieses Resultat ebenfalls für blinde Signaturverfahren.

Ferner wird bewiesen, dass es keine *black-box* Konstruktionen von blinden Signaturen basierend auf *one-way* Funktionen gibt. Dieses Ergebnis wird dahin gehend verallgemeinert, dass nicht nur Ansätze basierend auf einer zufälligen Permutation ausgeschlossen werden können, sondern auch Protokolle, die einen 1-bit Nachrichtenraum besitzen und nur Sicherheit gegen *honest-but-curious* Angreifer garantieren.

Contents

1	Introduction	1
2	Definitions	7
2.1	General Definitions	7
2.2	Blind Signature	7
2.3	Security of Blind Signature Schemes	8
3	On the Impossibility of Three-Move Blind Signature Schemes	11
3.1	Introduction	11
3.2	Hard Problems and Black-Box Reductions	16
3.3	Warm Up: Impossibility Result for Vanilla Reductions	18
3.4	Impossibility Result for Statistically Blind Signature Schemes	27
3.5	Impossibility Result for Computationally Blind Signature Schemes	40
4	Blind Signatures and Their Applications to Adaptive OT	49
4.1	Introduction	49
4.2	Selective-Failure Blindness	51
4.3	From Blindness to Selective-Failure Blindness	53
4.4	Selective Failures and Adaptive Oblivious Transfer	60
5	On the Impossibility of Blind Signatures From One-Way Permutations	69
5.1	Introduction	69
5.2	Overview of Our Techniques	70
5.3	Blind Signatures Relative to an Oracle	72
5.4	Attacking Black-Box Constructions of Blind Signatures	75
5.5	Extensions	84
	References	86

1 Introduction

Blind signature schemes, introduced by Chaum [Cha83a, Cha84], are a widely-studied primitive that allows a signer to interactively issue signatures for a user. Roughly, the signer learns nothing about the message being signed (*blindness*) while the user cannot compute any additional signatures without the help of the signer (*unforgeability*). Classical applications of blind signatures include e-cash, where a bank signs coins withdrawn by users, and e-voting, where an authority signs public keys that voters later use to cast their votes. In both scenarios, the signing party learns nothing about the coin (resp. the public key), while the receiver cannot create any additional signed coin (resp. public key).

Many blind signature schemes have been proposed in the literature, e.g., [Cha83b, JLO97a, PS00a, Abe01a, Bol03a, CKW04a, KZ05, Fis06a, Oka06a, HKKL07a, FS09, AO09], with varying characteristics of security and efficiency. Arguably, the most prominent example is the scheme by Chaum [Cha83b] shown in Figure 1.1. The basic idea of the protocol is to blind the hash $H(m)$ of the message m using a random value r^e . The user then receives the value $rH(m)^d$ and simply divides out the value r obtaining the signature $H(m)^d$ on the message m .

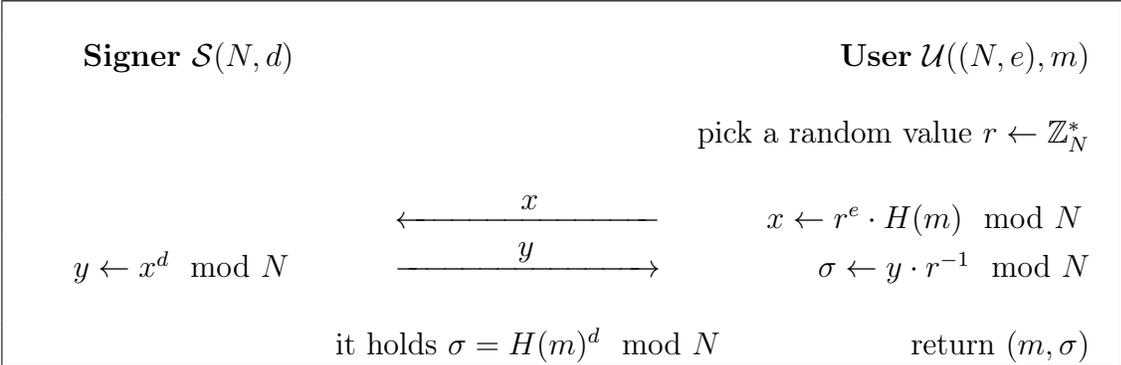


Figure 1.1: Chaum’s blind signature scheme using an RSA key-pair $(sk, pk) = ((N, d), (N, e))$, a message m , and a hash function H .

The various blind signature schemes differ in round complexity, underlying computational assumptions, and the model in which the security proof is given. For example,

many schemes [PS00b, Abe01b, BNPS03a, Bol03b, AO09] rely on the random oracle heuristic [BR93], where a hash function is considered as a truly random function. It is well-known, however, that a security proof in the random oracle model does not necessarily imply security in the standard model when a random oracle is instantiated by an efficient hash function [CGH98]. Therefore, alternative solutions are necessary. Several blind signature schemes that achieve security in the standard model have also been proposed [CKW04b, Oka06b, HK07, KZ08, AFG⁺10]. These instantiations differ in the underlying number-theoretic assumptions and their round complexities. Constructions based on general assumptions are also known [JLO97b, Fis06b, HKKL07b, FS09], but the *minimal assumptions* in terms of round complexity and computational assumptions without assuming setup assumptions are unknown.

Black-box Constructions and Reductions. In cryptography, most constructions are based on simpler building blocks, called primitives, that are believed to be secure. The underlying primitives are usually treated as a *black-box* in the sense that the construction does not make any assumptions about its concrete realization. To show that the construction is also secure, mainly black-box reduction techniques are applied. The initial assumption is the insecurity of the construction. The reduction then uses an algorithm that breaks the construction to solve the underlying primitive that was believed to be secure. This, however, is a contradiction and thus, our assumption must have been false. The security reduction is also black-box if it treats the underlying primitive and/or the adversary as a black-box. For a more formal approach see the paper of Reingold et al. [RTV04a]. The authors define different “flavors” of black-box reductions depending on the “degree” of black-box access to the primitive and/or adversary. Clearly, black-box constructions give only a limited view of the relation between different primitives as no conclusions about non-black-box access can be made. Nevertheless, this approach is well established, as most of the cryptographic proofs are black-box and it is powerful enough to show the equivalence between all primitives of private-key cryptography [Lam79, Lub92, IL89, ILL89, GL89, HILL99b, HRV10] including digital signature schemes [NY89, Rom90].

Black-box Separations. In some cases, no known black-box reductions exist between primitives. This raises the question if such constructions exist, or if there is a gap between the primitives. Impagliazzo and Rudich show in their seminal work that there indeed exists a black-box separation between primitives [IR89]. The basic idea is to demonstrate a world relative to which one of the primitives exists and the other does not. This approach was used by [IR89] to show that there is no construction of a secure key-agreement protocol (KA) based on one-way functions. This breakthrough result basically defines two separate worlds into which cryptographic primitives can

be divided: the “private-key” world that contains all those primitives that are equivalent to one-way functions (OWFs), such as digital signatures (DS), pseudorandom generators (PRGs), pseudorandom functions (PRFs), and private-key encryption, and the “public-key” world that contains primitives such as trapdoor permutations, public-key encryption (PKE), key-agreement (KA), private information retrieval (PIR), and oblivious transfer (OT). Figure 1.2 shows the separations between the main primitives (although many more separations are known).

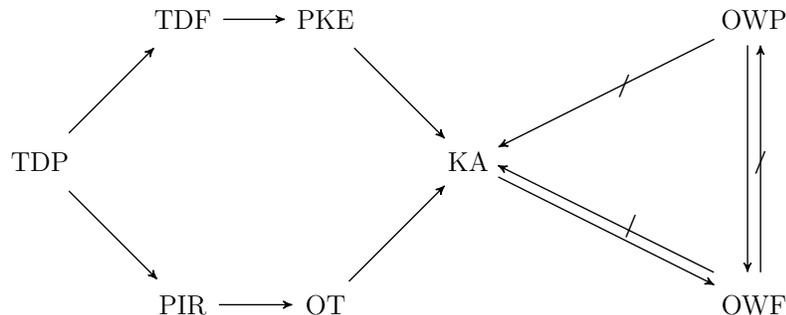


Figure 1.2: This figure shows the relations among the different primitives [CHL06]. Arrows signify black-box reductions, whereas crossed arrows denote black-box separations.

This technique has also been used to prove many other separation results, e.g., [Rud92, Sim98, KST99, GKM⁺00, GT00, GMR01, Fis02].

The second approach for showing separation between two primitives are meta-reductions: the basic idea is to “build a reduction against the reduction” [BV98, Cor02a, Bro06, PV06b, Bro07, BMV08a]. The assumption that there exists a reduction \mathcal{R} outlines the starting point. By assumption, the reduction \mathcal{R} with black-box access to any successful adversary \mathcal{A} breaks the underlying cryptographic problem. The difficulty is now to design an algorithm that simulates the reduction and that simultaneously mimics an adversary that “looks” good enough for the reduction. Such that at the end, the combination of the reduction and meta-reduction solve the underlying problem directly. This approach has been used to consider, for example in [PV06a, Bro07, BMV08b], the impossibility of reductions from secure encryption or signatures to a given RSA instance.

Contributions of this Thesis

In this thesis, we address both the round complexity and the minimal computational assumptions for constructing blind signature schemes. We begin by giving the foundations of blind signatures including the used notations in Chapter 2.

Chapter 3 then deals with the round complexity of blind signatures in the standard model. We prove the following theorem:

First Theorem. *There is no black-box reduction from the unforgeability of a three-move blind signature scheme (with signature-derivation checks) to any hard non-interactive computational problem.*

First, we consider the large class of three-move blind signatures where blindness holds in a statistical sense, i.e., where even an unbounded malicious signer cannot link executions of the issuing protocol to message/signature pairs; and where signature-derivation checks exist, i.e., one can verify from the communication between both parties if the honest user is able to derive a valid signature from the interaction. We then extend our result to computationally-blind signature schemes. To prove this result, we have to make some assumptions about the scheme. These assumptions basically came “for free” in the statistical case. In particular, we must assume that unforgeability and blindness of the scheme are somewhat independent. We omit the detail here and refer the reader to Section 3.1.3 for a comprehensive discussion.

In Chapter 4 we take the first action towards identifying the minimal computational assumption for blind signatures. We extend the idea of Camenisch et al. [CNS07] who show how to construct an adaptive oblivious transfer protocol out of any unique selective-failure blind signature scheme (in the random oracle model). Roughly speaking, *selective-failure blindness* says that

- (1) a malicious signer \mathcal{S}^* cannot force the user algorithm \mathcal{U} to abort based on the specific message and
- (2) blindness should also hold even if the signer is able to learn that some executions have aborted.

Uniqueness means that each message has only one signature per public key.

Obviously, if we manage to remove both additional properties (selective-failure blindness and uniqueness), we would immediately obtain a separation between blind signatures and OWF. This follows from the well-known result that oblivious transfer cannot be constructed from one-way functions in a black-box fashion. Therefore, our first step is to determine the minimal assumptions for a general transformation that turns any blind signature scheme into a selective-failure one. Our transformation, however, is not applicable to the construction of Camenisch et al. [CNS07] because it destroys the uniqueness of the blind signature scheme. In Section 4.4.2, we show that it is nonetheless possible to build an adaptive k -out-of- N oblivious transfer protocol out of *any* unique blind signature scheme by applying our transformation. The proposed construction is a modification of the protocol in [CNS07] and, because of the problems with uniqueness,

we have to prove the security of this construction from scratch.

Second Theorem. *If unique blind signature scheme exists in the random oracle model, then there exists simulatable adaptive oblivious transfer.*

At first glance, it seems that our protocol proves that blind signature cannot be constructed from one-way functions. This impression, however, is not quite true because the construction is based on a very strong assumption: *uniqueness*. In particular, in [FS10] we show that unique signatures cannot be constructed from trapdoor permutations in a black-box fashion. Yet, this does not say anything about the ability of constructing non-unique blind signature.

Nevertheless, in Chapter 5, we settle this question in the negative:

Third Theorem. *There is no black-box construction of a blind signature scheme from one-way functions.*

Our result imposes no restrictions on the blind signature scheme and applies even to schemes with imperfect completeness. Moreover, it is actually quite a bit more general than the above theorem indicates. In particular, our impossibility result holds also for constructions based on one-way permutations or random oracles, and even rules out constructions of blind signature schemes for 1-bit messages that achieve security only against honest-but-curious parties.

2 Definitions

2.1 General Definitions

Throughout this thesis, $n \in \mathbb{N}$ denotes the security parameter. Every algorithm A is a probabilistic Turing machine that runs in time polynomial in n unless indicated otherwise. Informally, we say that a function is *negligible* if it vanishes faster than the inverse of any polynomial. We usually refer to such a function as $\text{negl}(n)$. If S is a set, then $x \leftarrow S$ indicates that x is chosen uniformly at random over S (which in particular assumes that S can be sampled efficiently). We remark that, even if occasionally not mentioned, all algorithms in this thesis receive the security parameter 1^n as an additional input.

2.2 Blind Signature

To define blind signatures formally, we introduce the following notation for interactive execution between algorithms \mathcal{X} and \mathcal{Y} . By $(a, b) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ we denote the joint execution, where x is the private input of \mathcal{X} , y defines the private input for \mathcal{Y} , the private output of \mathcal{X} equals a , and the private output of \mathcal{Y} is b . We write $\mathcal{Y}^{\langle \mathcal{X}(x), \cdot \rangle^\infty}(y)$ if \mathcal{Y} can invoke an unbounded number of executions of the interactive protocol with \mathcal{X} in sequential order. Accordingly, $\mathcal{X}^{\langle \cdot, \mathcal{Y}(y_0) \rangle^1, \langle \cdot, \mathcal{Y}(y_1) \rangle^1}(x)$ can invoke sequentially ordered executions with $\mathcal{Y}(y_0)$ and $\mathcal{Y}(y_1)$, but interact with each algorithm only once.

Definition 2.2.1 (Blind Signature Scheme). *A blind signature scheme consists of a tuple of efficient algorithms $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ where*

KEY GENERATION. *The algorithm $\text{KG}(1^n)$ generates a key pair (sk, pk) .*

SIGNATURE ISSUING. *The joint execution of algorithm $\mathcal{S}(sk)$ and algorithm $\mathcal{U}(pk, m)$ for message $m \in \{0, 1\}^n$ generates an output σ of the user, $(\perp, \sigma) \leftarrow \langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$, where possibly $\sigma = \perp$.*

VERIFICATION. *The algorithm $\text{Vf}(pk, m, \sigma)$ outputs a bit b , indicating whether the signature is valid or not.*

It is assumed that the scheme is complete, i.e., for any $(sk, pk) \leftarrow \text{KG}(1^n)$, any message $m \in \{0, 1\}^n$ and any σ output by \mathcal{U} in the joint execution of $\mathcal{S}(sk)$ and $\mathcal{U}(pk, m)$ we have $\text{Vf}(pk, m, \sigma) = 1$.

We also consider blind signature schemes that have *imperfect* completeness:

Definition 2.2.2 (Imperfect Completeness). *A blind signature scheme BS has imperfect completeness if with overwhelming probability in $n \in \mathbb{N}$ the following holds: For $(sk, pk) \leftarrow \text{KG}(1^n)$, any message $m \in \{0, 1\}^*$ and any σ output by \mathcal{U} in the joint execution of $\mathcal{S}(sk)$ and $\mathcal{U}(pk, m)$ we have $\text{Vf}(pk, m, \sigma) = 1$.*

2.3 Security of Blind Signature Schemes

Security of blind signature schemes requires two properties, unforgeability and blindness [JLO97b, PS96].

2.3.1 Unforgeability

A malicious user \mathcal{U}^* against unforgeability tries to generate $k + 1$ valid message-signatures pairs after at most k completed interactions with the signer, where the number of interactions is adaptively determined by the user during the attack. The blindness condition says that it should be infeasible for a malicious signer \mathcal{S}^* to decide upon the order in which two messages m_0 and m_1 have been signed in two executions with an honest user \mathcal{U} .

Definition 2.3.1 (Unforgeability). *A blind signature scheme $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ is unforgeable if for any efficient algorithm \mathcal{U}^* the probability that the experiment $\text{Forge}_{\mathcal{U}^*}^{\text{BS}}$ evaluates to 1 is negligible (as a function of n), where*

Experiment $\text{Forge}_{\mathcal{U}^*}^{\text{BS}}(n)$:

$(sk, pk) \leftarrow \text{KG}(1^n)$

$((m_1, \sigma_1), \dots, (m_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{U}^{*(\mathcal{S}(sk), \cdot)^\infty}(pk)$

Return 1 iff

$m_i \neq m_j$ for i, j with $i \neq j$, and

$\text{Vf}(pk, m_i, \sigma_i) = 1$ for all $i = 1, 2, \dots, k + 1$, and

at most k interactions with $\langle \mathcal{S}(sk), \cdot \rangle^\infty$ were completed.

2.3.2 Computational and Statistical Blindness

Blindness says that it should be infeasible for a malicious signer \mathcal{S}^* to decide which of two messages m_0 and m_1 was signed first in two executions with an honest user \mathcal{U} . In general, this condition must hold, even if \mathcal{S}^* is allowed to choose the public key maliciously.

Definition 2.3.2 (Blindness). *A blind signature scheme $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ is computational blind (resp. statistical blind) if for any (efficient resp. unbounded) algorithm \mathcal{S}^* working in modes *find*, *issue* and *guess*, the probability that the following experiment $\text{Blind}_{\mathcal{S}^*}^{\text{BS}}$ evaluates to 1 is negligibly close to $1/2$, where*

Experiment $\text{Blind}_{\mathcal{S}^*}^{\text{BS}}(n)$:

$(pk, m_0, m_1, \text{st}_{\text{find}}) \leftarrow \mathcal{S}^*(\text{find}, 1^n)$

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

$\text{st}_{\text{issue}} \leftarrow \mathcal{S}^*(\langle \cdot, \mathcal{U}(pk, m_b) \rangle^1, \langle \cdot, \mathcal{U}(pk, m_{1-b}) \rangle^1)(\text{issue}, \text{st}_{\text{find}})$

and let σ_b, σ_{1-b} denote the (possibly undefined) local outputs of $\mathcal{U}(pk, m_b)$ resp. $\mathcal{U}(pk, m_{1-b})$.

set $(\sigma_0, \sigma_1) = (\perp, \perp)$ if $\sigma_0 = \perp$ or $\sigma_1 = \perp$

$b^* \leftarrow \mathcal{S}^*(\text{guess}, \sigma_0, \sigma_1, \text{st}_{\text{issue}})$

return 1 iff $b = b^*$.

Definition 2.3.3 (Secure Blind Signature Scheme). *A blind signature scheme $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ is secure if it is unforgeable and blind.*

3 On the Impossibility of Three-Move Blind Signature Schemes

3.1 Introduction

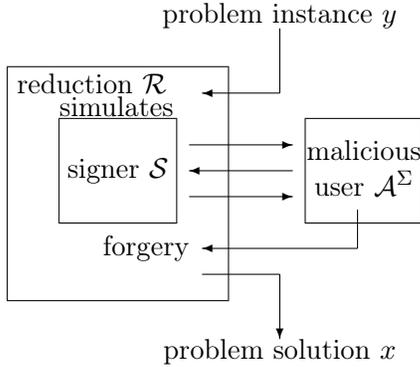
As already mentioned, the arguably most prominent examples of blind signatures are the blind signature schemes by Chaum [Cha83b] based on RSA and the ones by Pointcheval and Stern [PS00a] based on the discrete logarithm problem, RSA, and factoring. Both approaches admit a security proof in the random oracle model, in the case of Chaum’s scheme the “best” known security proofs currently even requires the one-more RSA assumption [BNPS03b].

In this chapter we investigate the possibility of instantiating the random oracles in the schemes by Chaum and by Pointcheval and Stern, and of giving a security proof based on standard assumptions like RSA or discrete logarithm. Although both schemes are different in nature we can subsume them under a more general pattern of blind signature schemes, where

- blindness holds in a statistical sense, i.e., even an unbounded malicious signer cannot link executions of the issuing protocol to message-signature pairs,
- the interactive signature issuing has three (or less) moves, and
- one can verify from the communication between a possibly malicious signer and an honest user if the user is eventually able to derive a valid signature from the interaction.

We note that the construction by Boldyreva [Bol03a] based on the one-more Gap Diffie-Hellman problem in the random oracle model also obeys these three properties such that any impossibility result immediately transfers to this scheme as well. The third property, which we coin *signature-derivation check*, basically guarantees that blindness still holds if the user fails to produce a signature in the postprocessing step, after the actual interaction with the signer has been completed. Common notions of blindness do not provide any security guarantee in this case (see Chapter 4 for detailed discussions).

$\mathcal{R}^{\mathcal{A}^\Sigma}$ SOLVES PROBLEM (IN PRESENCE OF Σ):



$\mathcal{M}^{\mathcal{R}}$ SOLVES PROBLEM (WITHOUT Σ):

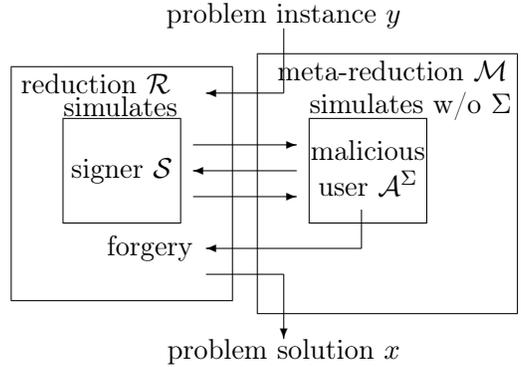


Figure 3.1: Meta-reduction technique: The black-box reduction \mathcal{R} on the left hand side uses the adversary \mathcal{A}^Σ against unforgeability to solve an instance y of the non-interactive problem. The meta-reduction \mathcal{M} on the right hand side then uses \mathcal{R} to solve the problem from scratch, i.e., by simulating \mathcal{A}^Σ without Σ . For this, the meta-reduction \mathcal{M} exploits the blindness property of the scheme.

3.1.1 The Idea Behind our Result

Given a blind signature scheme with the properties described above we can show that for such schemes finding black-box reductions from successful forgers to *any* underlying non-interactive cryptographic problem (like RSA, discrete-log, general one-wayness, or collision-resistance) is infeasible. The key idea to our result is as follows. Assume that we are given a three-move blind signature scheme as mentioned above and a reduction \mathcal{R} reducing unforgeability to a presumably hard problem (given only black-box access to an alleged forger). Vice versa, if the problem is indeed infeasible, then the reduction therefore shows that the scheme is unforgeable.

Our approach is to show that the existence of a reduction \mathcal{R} as above already violates the assumption about the hardness of the underlying problem. Our starting point is to design an oracle Σ with unlimited power and a “magic” adversary \mathcal{A}^Σ breaking the unforgeability of the blind signature scheme with the help of Σ . By assumption, the reduction \mathcal{R} with access to \mathcal{A}^Σ is then able to break the underlying cryptographic problem (see the left part of Figure 3.1). Note that, at this point, we are still in a setting with an all-powerful oracle Σ and the non-interactive problem may indeed be easy relative to this oracle, without contradicting the presumed hardness in the standard model.

Now we apply meta-reduction techniques, as put forward for example in [BV98, Cor02b, Bro06, PV06b], to remove the oracle Σ from the scenario. Given \mathcal{R} we show how to build a meta-reduction \mathcal{M} (a “reduction for the reduction”) to derive an efficient solver

for the problem, but now without any reference to the magic adversary and Σ (right part of Figure 3.1). To this end, the meta-reduction \mathcal{M} fills in for adversary \mathcal{A}^Σ and simulates the adversary’s actions without Σ , mainly by resetting the reduction \mathcal{R} appropriately. Then we have eventually derived an algorithm $\mathcal{M}^\mathcal{R}$ solving the underlying non-interactive problem in the standard model, meaning that the problem cannot be hard. In other words, there cannot exist such a reduction \mathcal{R} to a hard problem.¹

At this point it seems as if we have not used the blindness property of the scheme and that the idea would paradoxically also apply to regular signature schemes (for which we know secure constructions based on any one-way function). This is not the case. The blindness subtly guarantees that the meta-reduction’s simulation of the adversary is indistinguishable from the actual behavior of \mathcal{A}^Σ , such that the success probabilities of $\mathcal{R}^{\mathcal{A}^\Sigma}$ and of $\mathcal{M}^\mathcal{R}$ are close. For these two cases to be indistinguishable, namely \mathcal{R} communicating with \mathcal{A}^Σ or with \mathcal{M} , we particularly rely on the fact that blindness holds relative to the all-powerful oracle Σ used by \mathcal{A} , as in case of statistically-blind signature schemes.

3.1.2 The Essence of our Meta-Reduction and Impossibility of Random Oracle Instantiations

There are essentially two approaches in the literature to derive black-box separations like ours. One class of black-box separation results (e.g., [IR89, Sim98, RTV04b]) basically starts with an oracle Σ breaking any cryptographic primitive of type A , like a collision-resistant hash function, but adds an oracle Π implementing another primitive of type B like a one-way function (and which cannot be broken by Σ). Here, the cryptographic primitives in question are usually treated as black-boxes.

The other approach uses meta-reductions [BV98, Cor02a, Bro06, PV06b, Bro07, BMV08a] and usually treats the adversary as a black-box. In our case, we show that no black-box reduction to *arbitrary* (non-interactive) cryptographic problems can exist. This includes common assumptions like the RSA and discrete logarithm problem, but also more general notions of one-way functions and collision-resistant hash functions. Compared to oracle-based separations and previous meta-reduction techniques our result gives the following two advantages:

¹We consider very general reductions running *multiple* instances of the adversary in a concurrent and *resetting* manner, covering all known reductions for blind signatures in the literature. Yet, since the meta-reduction itself uses rewinding techniques, we somewhat need to restrict the reduction in regard of the order of starting and finishing resetted executions of different adversarial instances (called resetting with restricted cross-resets). This saves us from an exponential running time for \mathcal{M} . For example, any resetting reduction running only a single adversarial instance at a time obeys our restriction.

- Oracle separations involving a “positive” oracle Π implementing a primitive often do not allow to make statements about the possibility of deriving schemes based on concrete primitives such as RSA or discrete-log. The latter primitives have other properties which could potentially be exploited for a security proof, like homomorphic properties. This limitation does not hold for our results.
- Meta-reduction separations such as [PV06b, Bro07, BMV08a] consider the impossibility of reductions from secure encryption or signatures to a given RSA instance. Yet, they often fall short of providing any meaningful claim if other assumptions enter the security proof, e.g., the result in [PV06b] does not hold anymore if two RSA instances are given or an additional collision-resistant hash function is used in the design. In comparison, our general approach covers such cases as we can easily combine non-interactive problems P_1, P_2 into more complex problems like $P_1 \vee P_2$ and $P_1 \wedge P_2$, requiring to break one of the two problems and both of them, respectively.

The latter advantage emerges because our meta-reduction plays off unforgeability against blindness. This technique may be useful in similar settings where two or more security properties are involved to provide stronger separation results for meta-reductions.

The broader class of problems ruled out by our meta-reduction also allows to make meaningful claims when it comes to the possibility instantiating the random oracle in the blind signature schemes. Namely, our separation indicates the limitations of hash function options (assuming some restriction on the resets of the reductions, mentioned in the previous section):

Any hash function whose security can be proven by black-box reduction to hard non-interactive problems does not allow a black-box reduction from the unforgeability of the blind signature scheme to hard non-interactive problems, such as RSA or discrete-logarithm.

This can be seen as follows. Any reduction from the unforgeability either breaks the underlying non-interactive problem like RSA or discrete-log, or breaks some security property of the hash function. The latter, in turn, yields a nested reduction from the unforgeability of the blind signature scheme to the non-interactive problem on which the hash function is based. One only needs to ensure that this nested reduction falls within our admissible reset strategy. This is clearly true if the security property of the hash function is given by a hard non-interactive problem itself, like one-wayness or collision-resistance, or allows a suitable reduction to these problems or RSA, discrete-log etc.

3.1.3 Extension to Computational Blindness

In principle, our result extends to computationally-blind signature schemes but the conditions are arguably more restrictive than in the statistical case. First, recall that blindness needs to hold relative to the forgery oracle Σ , i.e., the powerful forgery oracle must not facilitate the task of breaking blindness. While this comes “for free” in the statistical case, in the computational case one must assume that unforgeability and blindness of the scheme are somewhat independent. This is true for instance for Fischlin’s scheme [Fis06a], but there are also examples where blindness and unforgeability are correlated, as in Abe’s scheme [Abe01a] where unforgeability is based on the discrete-log problem and blindness on the DDH problem.

Second, given that the scheme is computationally-blind relative to Σ we still rely on the signature derivation check. One can easily design computationally-blind schemes infringing this property, say, by letting the user send a public key and having the signer encrypt each reply (we are not aware of any counter example in the statistical case). On the other hand, these signature derivation checks are very common, e.g., besides the schemes above the ones by Okamoto [Oka06a] and by Fischlin [Fis06a] too have this property.

Third, since we have to change the forgery oracle Σ for the computational case, we also need a key-validity check which allows to verify if a public key has a matching secret key (i.e., if there is a key pair with this public key in the range of the key generating algorithm). For schemes based on discrete-logarithm this usually boils down to check that the values are group elements. Given that these three conditions are met we show that our techniques carry over to the computational case.

Related Work. In a sense, our results match the current knowledge about the round complexity of blind signature schemes. Nowadays, the best upper bound to build (non-concurrently) secure blind signatures are four moves for the standard model, i.e., neither using random oracles nor set-up assumptions like a common reference string. This is achieved by a protocol of Okamoto [Oka06a] based on the 2SDH bilinear Diffie-Hellman assumption. Any schemes with three moves or less either use the random oracle model [Cha83b, PS00a, Bol03a] or a common reference string [Fis06a, HK07, AO09, AFG⁺10].

We note that Lindell [Lin03] rules out any concurrently secure blind signature scheme in the standard model, independently of any cryptographic assumption. Hence, it seems that two-move schemes—which are concurrently secure by nature—are impossible in the standard model. However, Lindell’s impossibility result only refers to the stronger (black-box) *simulation-based* definition of blind schemes and can indeed be circumvented by switching to the common *game-based* definition, as shown by [HKKL07a].

In contrast, our result holds with respect to game-based definitions and also covers three-move schemes, thus showing that such blind signature schemes may be hard to build even under this relaxed notion.

The recent results by Brown [Bro07] and Bresson et al. [BMV08a] show meta-reduction based separations of the one-more RSA and one-more discrete-logarithm problem from their regular counterparts. The conclusion in [BMV08a] is that it should be hard to find a security proof for Chaum’s scheme and the Pointcheval-Stern schemes using only these regular assumptions. As mentioned before, the meta-reductions in [Bro07, BMV08a] are limited in the sense that they either cannot rewind (as in [Bro07]) or can only forward the input RSA or discrete log problem (as in [BMV08a]). Our approach, however, considers arbitrary hard non-interactive problems and is robust with respect to the combination of several underlying assumptions.

We also remark that the well-known three-move lower bound for non-trivial zero-knowledge [GK96] is not known to provide a lower bound for blind signature schemes. The intuitively appealing idea of using the blind signature scheme as a commitment scheme in such zero-knowledge proofs unfortunately results in proofs which require more than three moves. This is even true if we start with a two-move blind signature scheme where a “hidden” third move is required for the initial transmission of the signer’s public key. In addition, the game-based notion of blind signatures is not known to yield appropriate zero-knowledge simulators.

3.2 Hard Problems and Black-Box Reductions

In order to prove the security of a cryptographic protocol, usually reduction techniques are used. A reduction from a cryptographic protocol to an underlying problem shows that breaking the protocol implies breaking the underlying problem. A reduction is *black-box* if it treats the adversary and/or the underlying primitive as an oracle. Reingold et al. [RTV04b] call reductions which use both the adversary and the primitive merely as an oracle *fully-black-box*, whereas *semi-black-box* reductions work for any efficient adversaries (whose code the reduction may access) as long as the primitive is black-box.

In our case we only need the orthogonal requirement to semi-black-box reductions, namely the reduction treats the adversary as an oracle but we do not make any assumption about the representation of the underlying primitive. The reduction we consider works for any kind of non-interactive primitive (i.e., in which one gets an instance as input and outputs a solution without further interaction):

Definition 3.2.1 (Hard Non-Interactive Problem). *A non-interactive (cryptographic) problem $P = (I, V)$ consists of two efficient algorithms:*

INSTANCE GENERATION. *The instance generation algorithm $I(1^n)$ takes as input the security parameter 1^n and outputs an instance $y \leftarrow I(1^n)$.*

INSTANCE VERIFICATION. *The instance verification algorithm $V(x, y)$ takes as input a value x as well as an instance y of a cryptographic problem and outputs a decision bit $b \leftarrow V(x, y)$.*

We call a cryptographic problem P hard if the following condition is fulfilled:

HARDNESS. *We say that an algorithm \mathcal{A} solves the cryptographic problem P if the probability that \mathcal{A} on input $y \leftarrow I(1^n)$ outputs x' such that $V(x', y) = 1$, is non-negligible. We say that the problem P is hard if no efficient algorithm solves it.*

Note that in the definition mentioned above we do not impose any completeness requirement on the cryptographic problem. The reason is that reductions from the security of blind signatures must work for arbitrary problems and in particular to the ones with non-trivial completeness conditions.

The notion of a non-interactive cryptographic problem clearly covers such popular cases like the RSA problem, the discrete logarithm problem, or finding collisions for hash functions. It also comprises more elaborate combination of such problems, e.g., if P_0, P_1 are two non-interactive problems then so are $P_0 \wedge P_1$ and $P_0 \vee P_1$ (with the straightforward meaning requiring to solve both problems or at least one of them).

Very often in cryptography one builds protocols from several primitives P_0, P_1, \dots, P_k , and one gets a sequence of reductions $\mathcal{R}_1, \dots, \mathcal{R}_k$ to each primitive, but where the reduction \mathcal{R}_i has full control over the other primitives. For instance, a protocol may rely on the RSA problem (P_0) and collision-intractable hash functions (P_1) and any break either yields an RSA solver (\mathcal{R}_0) or a collision-finder (\mathcal{R}_1). But reduction \mathcal{R}_1 itself may take advantage of the fact that it knows the factorization for the RSA-part (or even picks the modulus itself). Such cases are also subsumed by considering the problem P which generates $y_i \leftarrow I_i(1^n)$ for $i = 1, 2, \dots, k$ and outputs a randomly chosen instance.

3.3 Warm Up: Impossibility Result for Vanilla Reductions

To give some intuition about our technique we first consider the simpler case of *vanilla* reductions. This type of reduction only runs a single execution with the adversary (without rewinding) and, if communicating with an honest user, makes the user output a valid signature with probability 1. This means that a vanilla reduction takes advantage of the magic adversary and its output, instead of solving the problem on its own. We then augment our result in the next section to deal with resetting reductions running multiple adversarial instances.

3.3.1 Preliminaries

For our impossibility result we need another requirement on the blind signature scheme, besides statistically blindness. This property says that one can tell from the public data and communication between a malicious signer and an honest user whether the user is able to compute a valid signature or not.

For instance, in Chaum’s scheme (see Figure 1.1 on Page 1) the honest user sends a value y and receives z from the signer, and the user is able to compute a signature σ for an arbitrary message m if and only if $z^e = y \bmod N$. This is easily verifiable with the help of the public key and the communication. The scheme of Pointcheval and Stern implements the signature-derivation check already in the user algorithm.² Analogous derivation checks occur in the schemes by Okamoto and by Fischlin. More formally:

Definition 3.3.1 (Signature-Derivation Check). *A blind signature scheme BS allows (computational resp. statistical) signature-derivation checks if there exists an efficient algorithm SDCh such that for any (efficient resp. unbounded) algorithm \mathcal{S}^* working in modes *find* and *issue* the probability that the experiment $\text{SigDerCheck}_{\mathcal{S}^*, \text{SDCh}}^{\text{BS}}$ evaluates to 1 is negligible, where*

Experiment $\text{SigDerCheck}_{\mathcal{S}^*, \text{SDCh}}^{\text{BS}}(n)$:

$(pk, m, st) \leftarrow \mathcal{S}^*(\text{find}, 1^n)$

$(\perp, \sigma) \leftarrow \langle \mathcal{S}^*(\text{issue}, st), \mathcal{U}(pk, m) \rangle$

where *trans* denotes the communication between \mathcal{S}^* , \mathcal{U}

$c \leftarrow \text{SDCh}(pk, \text{trans})$

return 1 if $\sigma \neq \perp$ and $c = 0$, or if $\sigma = \perp$ but $c = 1$.

²The signature derivation check is given by the user’s local verification $a = g^R h^S y^e$, where the values a, r, R, S are exchanged during the signature issuing protocol and the values g, h, y are part of the public key.

In the computational case, if the above holds even if \mathcal{S}^* gets access to an oracle Σ then we say that the scheme has computational signature-derivation checks relative to Σ . (In the statistical case \mathcal{S}^* could simulate Σ internally, such that granting access to Σ is redundant.)

The notion in some sense augments the blindness property of blind signature schemes to the case that the user algorithm fails to produce a valid signature in the final local step. The common notion of blindness does not provide any security in this case (because the malicious signer does not receive any of the signatures if the user fails only then). See [FS09] for more discussions and solutions. Here, the signature-derivation check provides something stronger, as it can be efficiently performed by anyone and holds independently of the user's message.

Next we introduce a weaker notion than blindness which is geared towards our separation result. Informally, a blind signature scheme has so-called *transcript-independent signatures* if one cannot associate a transcript to a signature. This is formalized by comparing signatures generated via an execution with a malicious signer and signatures generated “magically” via an oracle Σ producing the signature for a message from the public key and the transcript of the first execution. The intuition behind the following experiment is that the malicious signer has to distinguish whether the second signature σ_b results from the signature issuing protocol, or if the oracle Σ derived the signature σ_b from the transcript of the signature issuing protocol where the honest user gets as input the message m_0 .

Definition 3.3.2 (Transcript-Independent Signatures). *A blind signature scheme BS has (computationally resp. statistically) transcript-independent signatures with respect to Σ if for any (efficient resp. unbounded) algorithm \mathcal{S}_{trans}^* the probability that the experiment $\text{trans-ind}_{\mathcal{S}_{trans}^*, \Sigma}^{\text{BS}}(n)$ evaluates to 1 is negligibly close to 1/2, where*

Experiment $\text{trans-ind}_{\mathcal{S}_{trans}^*, \Sigma}^{\text{BS}}(n)$:

- $b \leftarrow \{0, 1\}$
- $(pk, st_1, m_{-1}, m_0) \leftarrow \mathcal{S}_{trans}^{*, \Sigma}(init, 1^n)$
- $st_2 \leftarrow \mathcal{S}_{trans}^{*, \Sigma}(\langle \cdot, \mathcal{U}(pk, m_{-1}) \rangle^1, \langle \cdot, \mathcal{U}(pk, m_0) \rangle^1)(issue, st_1)$
- let σ_{-1} and σ_0 be the local outputs of the users in the two executions (possibly $\sigma_{-1} = \perp$ and/or $\sigma_0 = \perp$)
- and let $trans_{-1}$ be the transcript of the left execution
- set $m_1 = m_0$ and compute $\sigma_1 \leftarrow \Sigma(pk, trans_{-1}, m_1)$
- set $(\sigma_{-1}, \sigma_0, \sigma_1) = (\perp, \perp, \perp)$ if $\sigma_{-1} = \perp$ or $\sigma_0 = \perp$ or $\sigma_1 = \perp$
- $b^* \leftarrow \mathcal{S}_{trans}^{*, \Sigma}(guess, st_2, m_{-1}, \sigma_{-1}, m_b, \sigma_b)$
- return 1 iff $b = b^*$.

To define our generic forgery oracle Σ allowing \mathcal{A} to break unforgeability we first outline the idea for the case of Chaum’s blind signature scheme. Assume that the adversary has already obtained a valid signature for some message m' by communicating with the signer. Let $\mathbf{trans} = (y, z)$ denote the transcript of this communication. Algorithm $\Sigma(pk, \mathbf{trans}, m)$ for $m \neq m'$ then searches some randomness r such that the user’s first message for m and r matches y in the transcript, i.e., $H(m)r^e \bmod N = y$. Such an r exists by the perfect blindness and the signature-derivation check.³

The above example can be generalized to any blind signature scheme and the following generic forgery oracle (which only depends on the blind signature scheme in question):

Definition 3.3.3 (Generic Forgery Oracle). *For a statistically-blind signature scheme BS the generic forgery oracle $\Sigma(pk, \mathbf{trans}, m)$ performs the following steps:*

*enumerate all values r such that
the user algorithm $\mathcal{U}(pk, m)$ for randomness r generates the same
transcript \mathbf{trans} when fed with the same signer messages as in \mathbf{trans} ;
also store all signatures σ the user’s algorithm generates in these executions.
select a value r of the set at random and return the corresponding signature σ
(or return \perp if there is no such r).*

The next proposition shows that every statistically blind signature scheme that allows signature-derivation checks which has access to Σ has already transcript-independent signatures.

Proposition 3.3.4. *Every statistically blind signature scheme, which has statistical signature-derivation checks, also has statistical transcript-independent signatures with respect to the generic forgery oracle Σ .*

Proof. Assume that there exists a signer $\mathcal{S}_{\mathbf{trans}}^*$ in experiment $\mathbf{trans}\text{-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ with the generic forgery oracle Σ which outputs $b^* = b$ with non-negligible probability beyond $1/2$. Then we construct an adversarial controlled signer $\mathcal{S}_{\text{blind}}^*$ against the blindness (with oracle access to Σ) as follows. Algorithm $\mathcal{S}_{\text{blind}}^*$ invokes $\mathcal{S}_{\mathbf{trans}}^*(\text{init}, 1^n)$ to get $(pk, \mathbf{st}_1, m_{-1}, m_0)$; it uses its access to Σ to answer any request of $\mathcal{S}_{\mathbf{trans}}^*$ to this oracle. Algorithm $\mathcal{S}_{\text{blind}}^*$ then outputs (pk, m_{-1}, m_0) according to the blindness experiment and subsequently relays the entire communication between the two honest user instances \mathcal{U} and $\mathcal{S}_{\mathbf{trans}}^*$. In the case that $\mathcal{S}_{\text{blind}}^*$ obtains two undefined signatures from the blindness experiment, i.e., $(\sigma_{-1}, \sigma_0) = (\perp, \perp)$, then $\mathcal{S}_{\text{blind}}^*$ returns $(m_{-1}, \perp, m_0, \perp)$ to

³Note that blindness of Chaum’s scheme is only guaranteed if the user can verify that the exponent e is relatively prime to $\varphi(N)$, say, if e is a prime larger than N ; only then is guaranteed that the function $(\cdot)^e \bmod N$ really is a permutation.

$\mathcal{S}_{\text{trans}}^*$. Otherwise, if both executions have been successful, then $\mathcal{S}_{\text{blind}}^*$ executes $\mathcal{S}_{\text{trans}}^*$ in mode `guess` on input $(\text{st}_2, m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ to obtain a bit b^* , where st_2 is the state returned by $\mathcal{S}_{\text{trans}}^*(\text{st}_1)$ after the interaction with the users. Algorithm $\mathcal{S}_{\text{blind}}^*$ returns b^* as its decisional bit.

For the analysis first observe that, if the left user instance yields a valid signature $\sigma_{-1} \neq \perp$, then Σ too succeeds in producing a valid signature with overwhelming probability. This is true since the scheme allows signature-derivation checks and is statistically blind. More specifically, call a tuple $(pk, \text{st}_1, m_{-1}, m_0)$ output by the transcript adversary $\mathcal{S}_{\text{trans}}^*$ *bad* if the probability (over the user's randomness) that the user is able to produce a valid signature from the communication with the transcript adversary for m_{-1} , but the signature-derivation check returns 0 or the transcript is not in the range of possible transcripts for message m_0 , is non-negligible. Note that we can assume that $\mathcal{S}_{\text{trans}}^*$ is deterministic and chooses the bad tuple $(pk, \text{st}_1, m_{-1}, m_0)$ that maximizes the probability. Then the probability that the signature-derivation check answers inconsistently is negligible. The probability that the transcript is not in the range for message m_0 is negligible by the statistical blindness (else one could easily break blindness with the help of $\mathcal{S}_{\text{trans}}^*$). It follows that there is no bad tuple.

Hence, given that the user picks randomness such that it can compute a signature, except with negligible probability the transcript is also in the range for m_0 and the signature-derivation check indicates success. Since the answer of the signature-derivation check only depends on the public key and the transcript, it follows that the user's algorithm is in principle also able to derive a signature for m_0 . Therefore, the forgery oracle is able to find such a valid signature, except with negligible probability. From now we can thus assume that the transcript adversary receives undefined signatures only if one of the user instances fails to compute a signature.

Consider now the case where the bit b equals 0. The adversary $\mathcal{S}_{\text{blind}}^*$ in this case receives the second message-signature pair (m_0, σ_0) from the right execution with the honest user \mathcal{U} . It is easy to see that this experiment corresponds (almost) exactly to the blindness experiment (taking into account that undefined signatures only depend on success in the user instances). Thus, $\mathcal{S}_{\text{blind}}^*$ performs an almost perfect simulation from $\mathcal{S}_{\text{trans}}^*$'s point of view.

Now we investigate the case $b = 1$. In the blindness experiment the malicious signer then communicates with the left user instance $\mathcal{U}(pk, m_1)$ and with the right instance $\mathcal{U}(pk, m_0)$. In contrast, $\mathcal{S}_{\text{trans}}^*$ in the transcript-independence experiment interacts on the left side with a user instance that has been initialized with the message m_0 (instead of m_{-1}) and obtains the second signature σ_{-1} from the oracle Σ for the same message $m_1 = m_0$.

It holds again that the oracle succeeds whenever the left user instance is able to compute a signature (this follows immediately by construction of Σ since the set of possible random inputs contains at least the actual randomness used by the honest user in the left instance). Because the forgery oracle enumerates all possible randomness r such that it can derive a valid signature and selects one of them at random, the output distribution here is identical to the case of an interaction with the user. Thus, the simulation of $\mathcal{S}_{\text{trans}}^*$ is perfect in this case. But then we can conclude that if $\mathcal{S}_{\text{trans}}^*$ succeeds with non-negligible probability over $1/2$, then $\mathcal{S}_{\text{blind}}^*$ also succeeds with non-negligible probability bounded away from $1/2$. \square

We point out that the proof implicitly shows that, if the left user instance in the transcript-independence experiment succeeds in producing a signature, then so does the generic forgery oracle with overwhelming probability. Since this will be used again later in the proof of the separation result we state this as a corollary more explicitly:

Corollary 3.3.5. *For every statistically blind signature scheme BS with statistical signature-derivation checks, which is blind relative to the generic forgery oracle Σ , the probability that in the transcript-independence experiment we have $\sigma_{-1} \neq \perp$ and $\sigma_1 = \perp$ after the run of Σ , is negligible.*

Given the generic forgery oracle Σ we can now define the “magic” adversary which first plays an honest users communicating with the signer once. If this single execution yields a valid signature (which is certainly the case when interacting with the genuine signer, but possibly not when interacting with the reduction), then the adversary generates another valid message-signature pair without interaction but using Σ as a subroutine instead.

Definition 3.3.6 (Magic Adversary). *The magic adversary \mathcal{A} for input pk and with oracle access to the generic forgery oracle Σ and communicating with an oracle $\langle \mathcal{S}(sk), \cdot \rangle^1$ is described by the following steps:*

*pick random messages $m'_0, m'_1 \leftarrow \{0, 1\}^n$
run an execution $\langle \mathcal{S}(sk), \mathcal{U}(pk, m'_0) \rangle$ in the role of an honest user
to obtain σ'_0 and let trans'_0 be the corresponding transcript
if $\mathbf{Vf}(pk, m'_0, \sigma'_0) = 1$ then let $\sigma'_1 \leftarrow \Sigma(pk, \text{trans}'_0, m'_1)$ else set $\sigma'_1 \leftarrow \perp$
return $(m'_0, \sigma'_0, m'_1, \sigma'_1)$*

By the completeness of the blind signature scheme the magic adversary, when attacking the honest signer, returns two valid message-signature pairs, with probability negligibly close to 1 (there is a probability of at most 2^{-n} that the adversary outputs identical

pairs for $m'_0 = m'_1$). We also remark that the magic adversary, when attacking the actual scheme, applies the forgery oracle to derive a signature for the second message using the transcript of the first signature issuing protocol.

3.3.2 Impossibility Result

The following theorem states that vanilla black-box reductions to (non-interactive) cryptographic problems do not provide a meaningful security statement. That is, if there was such a reduction then the underlying problem would already be easy. Since we only deal with non-resetting reductions the claim even holds for schemes with arbitrary round complexity (instead of three-move schemes):

Theorem 3.3.7. *Let BS be a statistically blind signature scheme that allows statistical signature-derivation checks. Then there is no vanilla black-box reduction from unforgeability of the blind signature scheme BS to a hard non-interactive problem.*

We first illustrate the proof idea of Theorem 3.3.7. Assume that there was such a reduction \mathcal{R} taking an instance y of the problem as input and consider the magic adversary. Unless the reduction is trivial and solves the problem without the help of this adversary—in which case we are already done—the reduction \mathcal{R} has to simulate the adversary once to get two message-signature pairs. In particular, this means that, with probability 1, the reduction provides “good” values from which an honest user can derive a valid signature.

We now construct a meta-reduction \mathcal{M} which mimics the adversarial behavior by rewinding the reduction \mathcal{R} after the first execution, and branching into another independent execution (letting the reduction stay oblivious about the rewinding). By this, the meta-reduction gets two valid message-signature pairs with non-negligible probability. Algorithm \mathcal{M} forwards these two pairs to \mathcal{R} . If the reduction returns a valid solution x' to y then so does \mathcal{M} . But \mathcal{M} does not rely on the magic adversary and therefore provides an efficient solver for the underlying problem.

The difference between the generation of the second signature by the magic adversary and the one by the meta-reduction is that the former is computed “out of the blue”, while the latter is computed with the help of the reduction itself. Potentially, the success of \mathcal{R} may depend on a “trap” laid out in the public key and that \mathcal{M} ’s strategy of running two executions may defuse this trap. By the transcript-independence of the signature, though, this difference is insignificant, and the reduction must also work if playing against \mathcal{M} . We now turn to the formal proof:

Meta-reduction $\mathcal{M}(y)$
let $(pk, \text{st}_{\text{init}}) \leftarrow \mathcal{R}(\text{init}, y)$
let $(\text{msg}_1, \text{st}_{\text{msg}_1}) \leftarrow \mathcal{R}(\text{msg}_1, \text{st}_{\text{init}})$
choose $m_0 \leftarrow \{0, 1\}^n$ | choose $m_1 \leftarrow \{0, 1\}^n$
let $(\text{msg}_2, \text{st}_{\text{msg}_2}^0) \leftarrow \mathcal{U}(\text{msg}_2, pk, m_0, \text{msg}_1)$ | let $(\text{msg}_2, \text{st}_{\text{msg}_2}^1) \leftarrow \mathcal{U}(\text{msg}_2, pk, m_1, \text{msg}_1)$
let $(\text{msg}_3, \text{st}_{\text{msg}_3}^0) \leftarrow \mathcal{R}(\text{msg}_3, \text{st}_{\text{msg}_1}, \text{msg}_2)$ | let $(\text{msg}_3, \text{st}_{\text{msg}_3}^1) \leftarrow \mathcal{R}(\text{msg}_3, \text{st}_{\text{msg}_1}, \text{msg}_2)$
let $\sigma_0 \leftarrow \mathcal{U}(\text{finish}, \text{st}_{\text{msg}_2}^0, \text{msg}_3)$ | let $\sigma_1 \leftarrow \mathcal{U}(\text{finish}, \text{st}_{\text{msg}_2}^1, \text{msg}_3)$
output $x' \leftarrow \mathcal{R}(\text{final}, \text{st}_{\text{msg}_3}^0, m_0, \sigma_0, m_1, \sigma_1)$

Figure 3.2: Meta-Reduction for Vanilla Reduction (three moves), where $\text{trans}_0 = (\text{msg}_1, \text{msg}_2, \text{msg}_3)$ denotes the transcript of the first execution.

Proof. For sake of readability we divide the reduction \mathcal{R} into steps, according to the black-box simulation of the magic adversary in which \mathcal{R} takes over the role of the signer: in mode **init** the reduction outputs the public key pk and in mode **msg i** the reduction creates the i -th protocol message msg_i of the signer. After getting the adversary's signatures σ_0, σ_1 in the post-processing step **final** the reduction outputs a putative solution x' for its input y . In each step the reduction also outputs some state information which is passed on to the next stage.

Analogously to the reduction \mathcal{R} we denote by **msg j** the step of the honest user \mathcal{U} which on input a public key pk , a message m and the previous message msg_i of the signer, outputs message msg_j sent to the signer. Likewise, in mode **finish** the user creates the signature from its state and the final message sent by the signer.

Description of the Meta-Reduction. The meta-reduction \mathcal{M} works as follows (see Figure 3.2 for the case of three moves). It gets as input an instance y of the problem. It starts to simulate the reduction \mathcal{R} on y to derive a public key pk as well as the first message msg_1 on behalf of the signer and a state st_{msg_1} . Algorithm \mathcal{M} first completes an instance of the signature issuing protocol with \mathcal{R} using the program of the honest user on input a random message m_0 from $\{0, 1\}^n$ and some randomness r . Afterwards, it selects another message m' from $\{0, 1\}^n$ at random together with some independent randomness r' and resets the reduction to the point where \mathcal{R} has returned the first message of the signature issuing protocol. As before, \mathcal{M} executes the honest user algorithm on m' using the randomness r' .

Now, if the meta-reduction obtains two valid signatures σ_0, σ_1 from both executions, then it hands the pairs $(m_0, \sigma_0), (m_1, \sigma_1)$ to the reduction which then outputs some x' . The meta-reduction returns x' and stops. For brevity we often write $\mathcal{R}^{\mathcal{M}}(y)$ for this interaction.

Analysis of the Meta-Reduction. The final step is to show that the reduction \mathcal{R} successfully outputs a solution x' , even if given the pairs from \mathcal{M} instead of receiving them from the magic adversary. For this it suffices to show that

$$\text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1 \mid \mathcal{M}]$$

is non-negligible. As outlined above, for this we exploit the transcript-independence of signatures.

Assume to the contrary that the reduction \mathcal{R} outputs a valid solution x' with non-negligible probability if \mathcal{R} receives two message-signature pairs $(m_0, \sigma_0), (m_1, \sigma_1)$ from the magic adversary,

$$\text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}}(y) : V(x', y) = 1 \mid \mathcal{A} \text{ magic}] \not\approx 0,$$

but succeeds only with negligible probability if the message-signature pairs are generated by \mathcal{M} :

$$\text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1 \mid \mathcal{M}] \approx 0.$$

Then we construct an adversary $\mathcal{S}_{\text{trans}}^*$ who breaks the transcript independence of signatures in experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$.

Description of Adversary $\mathcal{S}_{\text{trans}}^*$. Informally, the adversary relays the first execution between the reduction and the external user instance and resets to reduction afterwards to answer the second execution. Afterwards $\mathcal{S}_{\text{trans}}^*$ receives two message-signature pairs without knowing whether the second signature σ_0 has been derived from the signature issuing protocol or with the help of Σ . We then use the result of the reduction to distinguish this case.

More formally, the adversary $\mathcal{S}_{\text{trans}}^*$ generates an instance $y \leftarrow I(n)$ of a cryptographic problem P . It simulates \mathcal{R} in a black-box way, which for input y initially outputs a public key pk as well as the first message msg1 and some state information st_{msg1} . The algorithm $\mathcal{S}_{\text{trans}}^*$ selects two random message $m_{-1}, m_0 \in \{0, 1\}^n$ and outputs pk, m_{-1}, m_0 according to the transcript-independence experiment. It stores the first message (from \mathcal{R} to \mathcal{U}) and relays the communication between the reduction \mathcal{R} and the first external user instance $\mathcal{U}(pk, m_{-1})$. Then the adversary resets \mathcal{R} to the point where \mathcal{R} has returned msg1 and forwards the communication between \mathcal{R} and \mathcal{U} .

After having finished both executions $\mathcal{S}_{\text{trans}}^*$ receives two (valid) signatures (σ_{-1}, σ_0) and runs the reduction \mathcal{R} in mode **final** on input $(\text{st}_{\text{msg3}}^0, m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ to obtain a

putative solution x' of the cryptographic problem P . The final output of the adversary is $b^* \leftarrow V(x', y)$.

Analysis of $\mathcal{S}_{\text{trans}}^*$. For the analysis recall that the magic adversary, after a single interaction, outputs two message-signature pairs (with the help of Σ). In fact, taking the message-signature pairs (m_{-1}, σ_{-1}) of the first execution together with the message-signature pair (m_0, σ_0) derived from Σ in experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ corresponds exactly to the behavior of the magic adversary ($b = 0$). Here we take advantage of the fact that the second execution with the user cannot fail (and force the signatures to be undefined) by our assumption about the vanilla reduction always making the honest user derive a signature.

On the other hand, during the issuing protocol with the honest user \mathcal{U} , the adversary $\mathcal{S}_{\text{trans}}^*$ resets \mathcal{R} and uses in the second execution the prefix `msg1` (obtained during the signature generation of (m_{-1}, σ_{-1})) in experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$. Therefore the message-signature pairs $(m_{-1}, \sigma_{-1}), (m_b, \sigma_b)$ are computed in the same way as the meta-reduction \mathcal{M} does ($b = 1$). Note that the additional run of Σ in the transcript-independence experiment cannot make the three signatures invalid (except with negligible probability), because of the statistical blindness and the signature derivation checks. More specifically, the statistical blindness guarantees that the transcript generated with \mathcal{U} for message m_{-1} is (almost surely) also a potential transcript for $m_0 = m_1$ used by Σ . Furthermore, the signature derivation check tells us that, independently of the message, the transcript allows the user to derive a signature (such that Σ , too, will find a valid random string r for the simulated user with a valid signature). This fact is stated more formally in Corollary 3.3.5. For simplicity we neglect the small error for Σ returning an invalid signature in the analysis below. We obtain for the probability that $\mathcal{S}_{\text{trans}}^*$ outputs the right bit $b^* = b$:

$$\text{Prob}[b^* = b] = \frac{1}{2} + \frac{1}{2} \cdot (\text{Prob}[b^* = 1 \mid b = 1] - \text{Prob}[b^* = 1 \mid b = 0])$$

According to our construction, $b = 0$ corresponds to the case where the simulation mimics the behavior of the magic adversary, and $b = 1$ the setting involving the meta-reduction. Furthermore, the adversary $\mathcal{S}_{\text{trans}}^*$ returns $b^* = 1$ in the case that the reduction \mathcal{R} returns a valid solution x' of y . Hence,

$$\begin{aligned} & \text{Prob}[b^* = 1 \mid b = 1] - \text{Prob}[b^* = 1 \mid b = 0] \\ &= \text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}}(y) : V(x', y) = 1 \mid \mathcal{A} \text{ magic}] \\ & \quad - \text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1 \mid \mathcal{M}]. \end{aligned}$$

By assumption the difference is non-negligible (because the first probability is non-

negligible and we have assumed that the second probability is negligible). This, however, contradicts the transcript independence of signatures. \square

3.4 Impossibility Result for Statistically Blind Signature Schemes

Here we discuss more general reductions which may reset the adversary and run several nested executions with multiple copies of the adversary.

For simplicity, we model a single, resettable instance of the adversary as a sequence of identical copies of the adversary which cannot be reset. Whenever the reduction seeks to reset the adversary, we instead invoke the next copy and run it up to the reset point with the same messages as before.

More precisely, we assume that the reduction \mathcal{R} is an interactive Turing machine which communicates with a “scheduled pool” of q^2 Turing machines $\mathcal{A}_{i,j}$ for $i, j = 1, 2, \dots, q$ for some polynomial $q = q(n)$ (which is bounded by the running time of the reduction). In this $q \times q$ matrix each Turing machine $\mathcal{A}_{i,1}, \dots, \mathcal{A}_{i,q}$ in row i is initialized with the same random string, which is chosen independently for each row.

We assume that the reduction has full control over the flow of interactions but can only deliver the i -th message in an execution after the $(i - 1)$ -st message in this execution has been sent (where we assume that the first transmission also comprises the public key). Instead of resetting the adversary $\mathcal{A}_{i,j}$ the reduction then invokes the next column $\mathcal{A}_{i,j+1}$ in this row and sends the same messages as before up to the reset point (but the reduction can never go back to a previous column). We also assume for simplicity that the reduction finishes each execution in a row before proceeding to the next column (say, by sending \perp as the third message). The q rows therefore correspond to q independent, resettable instances of the adversary, and in each row there is at any time only one “active” column execution.

3.4.1 Preliminaries

To build our meta-reduction we will reset the reduction continuously. That is, whenever the reduction expects a forgery from an instance of the magic adversary, we freeze the scenario and branch into a loop in which the meta-reduction seeks a second valid message-signature pair. In order to avoid an exponential blow-up in the running time of such rewinding executions [DNS04], we consider slightly restricted reductions.

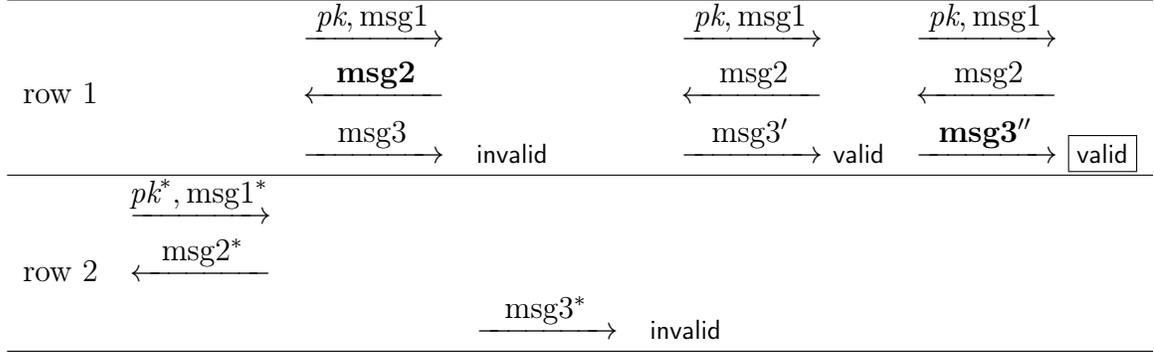


Figure 3.3: Example of a resetting scheduling with restricted cross-resets (executions in different rows may also run concurrently): Regarding the first and last execution in row 1 there is no other successful execution in between transmission of **msg2** and **msg3''**, except when it uses the same $(pk, \text{msg1})$ in the same row (as the third execution). The scheduling would violate the restricted resetting scenario if, for example, the execution in row 2 was valid (even if it was for the same $(pk^*, \text{msg1}^*) = (pk, \text{msg1})$), or if the third execution in row 1 was valid but for a different $(pk', \text{msg1}')$.

Resetting Reductions with Restricted Cross-Resets. Any reduction in our case is allowed to run concurrent executions with the copies of the adversary, each copy resetting at most q times, except that the reduction has to finish the interaction in the order according to the arrival of the second messages of the signature issue protocol. That is, suppose that the reduction receives the second message **msg2** (the user message) in some execution in row i which started with $(pk, \text{msg1})$. Suppose further that the reduction later finishes some execution with the same first transmission $(pk, \text{msg1})$ in the same row i by sending a third message allowing the user to derive a signature. Then, the reduction does not finish any other execution (in a different row or for distinct $(pk', \text{msg1}')$) in between these two points such that the user is also able to generate a valid signature for this execution (see Figure 3.3 for an example). By this, we can later rewind from the valid signature generation to the step where **msg2** has been sent, without destroying other executions which have been finished successfully meanwhile. Any reduction in our case is allowed to run $q = q(n)$ concurrent executions with the copies of the adversary, each copy resetting at most q times, except that the reduction has to finish the interaction in the order according to the arrival of the second messages of the signature issue protocol. That is, consider a three-move signature issuing run of the reduction with a copy of the adversary playing the honest user. Assume that the reduction receives the second message in this execution (which has been sent by the adversary resp. user), and call this execution pending from then on. We say that the reduction successfully finishes this pending execution if it sends the third message of the protocol such that the user is able to derive a valid signature.

The cross-reset restriction now demands that, if the reduction ever finishes a pending execution successfully, then there is no other execution which has become pending and has been finished successfully meanwhile. In other words, between the pending state of an execution and its completion the reduction may not receive the second message and complete any other execution (for which the user can compute a signature). We remark that the reduction may decide to entirely abort a pending execution and is still allowed to finish other pending executions, as long as the user is unable to produce a signature from that interaction.

Definition 3.4.1 (Resetting Reduction with Restricted Cross-Resets).

Consider a run of the reduction and its matrix of $q^2(n)$ executions. For an execution (i, j) let $\tau_{msg2}(i, j) = (t, pk, msg1)$ be the point in time t where the user message is delivered, and where $(pk, msg1)$ has been sent as the first transmission in this execution. Let $\tau_{valid}(i, j) = (t', pk', msg1')$ be the point in time t' , where the user receives the third message and is able to derive a valid signature from this interaction with first transmission $(pk', msg1')$. Then we say that the run of the reduction is resetting (with restricted cross-resets) if for any (i, j, k) such that $\tau_{msg2}(i, j) = (t, pk, msg1)$ and $\tau_{valid}(i, k) = (t', pk, msg1)$ for $t < t'$ the following holds: If there is (i^, j^*) such that $\tau_{valid}(i^*, j^*) = (t^*, pk^*, msg1^*)$ for $t < t^* < t'$ then $i^* = i$ and $(pk^*, msg1^*) = (pk, msg1)$. A reduction is called resetting with restricted cross-resets if it is resetting (with restricted cross-resets) for every run.*

Note that the scheduling of reductions with restricted cross-resets is related to so-called bounded concurrent (and resettable) executions [Bar01]. In m -bounded concurrent executions the number of instances running simultaneously is bounded by some fixed function $m = m(n)$ where the bound itself is known by the protocol. We do not put any a-priori bound on the number of concurrently running executions, because the number q of such instances depends on the reduction and is not bounded by a fixed polynomial. We merely restrict the way successful executions are finished. We also note that we can extend our proof below to allow a constant number of successfully finished executions between pending runs, but state and prove the simpler version for sake of readability.

q -wise Independent Hash Functions. An adequate measure to thwart reset attacks are usually pseudorandom functions (e.g., as in [CGGM00]). The idea is to make the randomness of the adversary depend on the communication by computing it as the output of the pseudorandom function for the communication. In this case, resetting the adversary essentially yields runs with independent random choices.

Here, we use the same idea but can fall back to the weaker requirement of q -wise independent hash functions in order to avoid the additional assumption that pseudorandom

functions exist. Roughly speaking, q -wise independent hash functions are functions that, when queried for q distinct preimages, output q independently distributed values:

Definition 3.4.2 (q -wise Independent Hash Function). *A family \mathcal{H} of efficiently computable functions $h : \{0, 1\}^a \mapsto \{0, 1\}^b$ is q -wise independent if for any distinct elements $x_1, x_2, \dots, x_q \in \{0, 1\}^a$ and any $y_1, \dots, y_q \in \{0, 1\}^b$ we have*

$$\text{Prob}[h(x_1) = y_1, \dots, h(x_q) = y_q] = (2^{-b})^q,$$

where the probability is taken over the random choice of h . We also assume that the sampling $h \leftarrow \mathcal{H}(1^n)$ is efficiently computable.

A typical example for q -wise independent hash function is a polynomial of degree $q - 1$ over $GF(2^a)$ for $a = b$.

We note that using q -wise independent hash functions instead of pseudorandom functions makes the adversary now depend on the reduction. Namely, below we use q as the number of maximal resets per row. However, since we deal with black-box reductions this is admissible. We also remark that we can overcome this dependency by using pseudorandom functions instead of q -wise independent hash function.

The New Magic Adversary. We use again the generic forgery oracle from the vanilla case. But here we augment our “new” magic adversary through a q -wise independent hash function (i.e., the random hash function h is given by parts of the adversary’s randomness). Informally, the adversary again runs the issuing protocol with the signer in the role of the honest user once. However, it now generates the message (and the user’s randomness) as the result of the q -wise independent hash function applied to the public key and the first message of the signer. Again, in the case that the single execution yields a valid signature, then the magic adversary here also creates another valid signature.

As we will later view Σ to be an integral part of the magic adversary and thus let the adversary provide the randomness $s \in \{0, 1\}^{\psi(n)}$ required by oracle Σ . We denote this augmented (deterministic) oracle with Σ^{aug} which now takes pk, trans, m and randomness s as input and returns σ . This randomness is also derived through the q -wise independent hash function, ensuring consistent answers for the same data ($pk, \text{msg}1$). We note that the length $\psi(n)$ of this randomness is only polynomial by construction of the generic forgery oracle:

Definition 3.4.3 (Magic Adversary). *The magic adversary $\mathcal{A} = \mathcal{A}_q$ (with parameter q) for input pk and access to the generic forgery oracle Σ^{aug} and communicating with an oracle $\langle \mathcal{S}(sk), \cdot \rangle^1$ works as described in the following steps:*

select a hash function h from the family of q -wise independent hash functions \mathcal{H} run an execution $\langle \mathcal{S}(sk), \mathcal{U}(pk, m'_0; r'_0) \rangle$ in the role of an honest user, where $(m'_0, m'_1, r'_0, s'_0) \leftarrow h(pk, msg1)$ is generated as the result of the q -wise independent hash function applied to the public key pk and the first message $msg1$ of \mathcal{S} ; let σ'_0 denote the resulting signature and $trans'_0$ the corresponding transcript.
if $\forall f(pk, m'_0, \sigma'_0) = 1$ then let $\sigma'_1 \leftarrow \Sigma^{aug}(pk, trans'_0, m'_1; s'_0)$ else set $\sigma'_0, \sigma'_1 \leftarrow \perp$
return $(m'_0, \sigma'_0, m'_1, \sigma'_1)$

It follows again from the completeness of BS together with the construction of the generic forgery oracle that the magic adversary succeeds in the unforgeability experiment with probability negligibly close to 1.

3.4.2 Impossibility Result

In the following we extend our result to restricted-cross resets.

Theorem 3.4.4. *Let BS be a three-move blind signature scheme, which is statistically blind and has statistical signature-derivation checks. Then there is no resetting (with restricted cross-resets) black-box reduction from unforgeability of the blind signature scheme BS to a hard non-interactive problem.*

Proof. The idea of the proof follows the one of Theorem 3.3.7 but differs in the point that the reduction \mathcal{R} is allowed to reset the adversary \mathcal{A} . In order to handle these resets, we provide the adversary with a q -wise independent hash function (i.e., we consider the adversary $\mathcal{A} = \mathcal{A}_q$). This makes each resetting execution independent and allows the meta-reduction \mathcal{M} to simulate the reduction. We can now also switch from Σ^{aug} to Σ as long as we guarantee that Σ gives identical answers for executions with the same $pk, msg1$; this can be easily implemented by table look-ups.

In the main step of the proof, we then construct a meta-reduction \mathcal{M} which mimics the adversarial behavior (without the help of Σ) by rewinding the reduction \mathcal{R} . This time, instead of rewinding the reduction only once in the only execution, our meta-reduction branches into a special loop phase to derive the second message-signature pair. Once entering this phase \mathcal{M} rewinds till it finds another accepting execution. Note that this is possible in polynomial time by our assumption about the restricted resetting scheduling, because no other execution is successfully finished meanwhile. When \mathcal{M} has found another valid pair it returns to the main simulation of the reduction.

We again show that \mathcal{M} 's behavior and the one of the magic adversary are indistinguishable to \mathcal{R} by the transcript independence of signatures. But this time, unlike in the

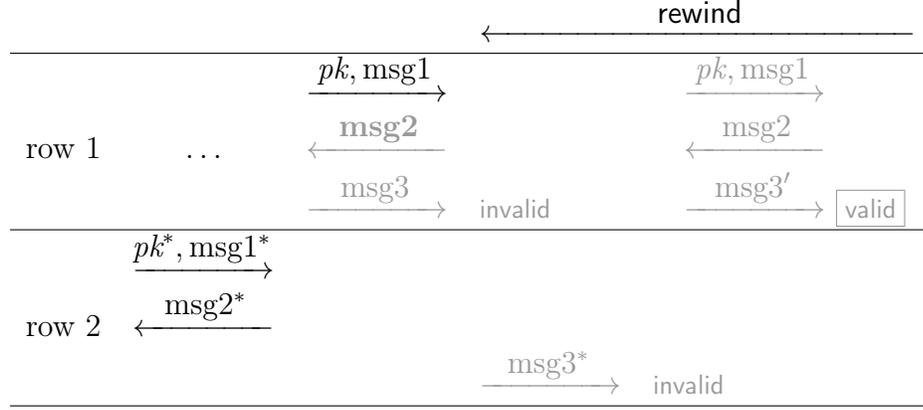


Figure 3.4: Continuously rewind to first execution in the row in which the same $(pk_{i,j}, \text{msg1}_{i,j})$ has been sent. By the restricted resetting scheduling no other execution can finish successfully meanwhile.

case of vanilla reductions where we only had a single rewinding, the meta-reductions here loops multiple times to find the second message-signature pair. In order to show that transcript independence guarantees indistinguishability in this case, we need to be able to check if we have inserted the external data from the transcript-independence experiment in the right loop. This can be ensured by the signature derivation checks.

Description of the Meta-Reduction. The input of the meta-reduction \mathcal{M} is an instance y of a cryptographic problem P . It runs black-box simulation of the reduction \mathcal{R} on input y and initializes an empty list L . This list stores elements of the form $(i, j, pk_{i,j}, \text{msg1}_{i,j}, m'_{i,j}, \sigma'_{i,j})$ which correspond to the (i, j) -th execution; where the tuple $(pk_{i,j}, \text{msg1}_{i,j})$ has been used during the first transmission; and the message-signature pair $(m'_{i,j}, \sigma'_{i,j})$ has been derived by rewinding.

Now, the reduction \mathcal{R} expects to communicate in a black-box way with an adversary \mathcal{A} . The meta-reduction \mathcal{M} mimics the magic adversary \mathcal{A} but computes the second message-signature pair differently. That is, consider the (i, j) -th execution, where the meta-reduction \mathcal{M} receives the third message $\text{msg3}_{i,j}$ that allows it to compute a signature $\sigma_{i,j}$ for a message $m_{i,j}$. The adversary, and thus the meta-reduction \mathcal{M} , is now supposed to output another valid message-signature pair. To do so, \mathcal{M} first checks whether it has already stored such a pair for the transmission $(pk_{i,j}, \text{msg1}_{i,j})$ in the list L , i.e., if $(i, h, pk_{i,j}, \text{msg1}_{i,j}, m'_{i,h}, \sigma'_{i,h}) \in L$ for some $h < j$. In this case, \mathcal{M} returns the tuple $(m_{i,j}, \sigma_{i,j})$ (which is the pair obtained through a “normal” execution) together with the pair $(m'_{i,h}, \sigma'_{i,h})$ (which is the pair derived by rewinding the reduction) to \mathcal{R} .

and continues the simulation. (This is consistent with the adversary's reply as in such rows the magic adversary too obtains identical answer from Σ^{aug} .)

Assume that \mathcal{M} does not find such an entry in L . The meta-reduction \mathcal{M} then searches through all communications in this row to find the first matching round $t \leq j$ where the adversary \mathcal{M} received $(pk_{i,j}, \text{msg1}_{i,j})$, i.e., it searches for the tuple $(t, pk_{i,j}, \text{msg1}_{i,j})$. It then freezes the simulation of \mathcal{R} and branches into a subroutine that executes a copy of \mathcal{R} for the same state before receiving the second message msg2 of the protocol in this execution, i.e., it rewinds the copy of \mathcal{R} to time t .

In the following we omit the index of the row since it is fixed and because it simplifies the notation. For the subprogram the meta-reduction repeats the following steps until \mathcal{M} is able to derive another message-signature pair. The meta-reduction \mathcal{M} keeps on rewinding the reduction (and thus the signature issuing protocol) to the point where the user algorithm \mathcal{U} computes the second message msg2_h for the h -th execution. For the ℓ -th rewinding, it selects an independent random message m_h^ℓ from $\{0, 1\}^n$ together with some randomness r_h^ℓ and continues the signature issuing protocol in the role of an honest user algorithm with \mathcal{R} . Observe that \mathcal{M} has rewound \mathcal{R} to the point where the user algorithm received $(pk_j, \text{msg1}_j)$, thus the first message and the public key remain unchanged. Since the reduction may have continued with other executions (a, b) , we use the same values $m_{a,b}, r_{a,b}$ as before in order to guarantee a consistent simulation. The meta-reduction starts with next loop if it does not find another valid pair in this execution, i.e., if this execution does not yield a valid pair for the same first transmission $(pk_j, \text{msg1}_j)$.

After \mathcal{M} has successfully derived a second message-signature pair (m'_h, σ'_h) in row i , it jumps back into the main execution (to the point where \mathcal{R} has sent the third message msg3 and the honest user algorithm has derived a valid message signature pair $(m_{i,j}, \sigma_{i,j})$), and returns both message-signature pairs $(m_{i,j}, \sigma_{i,j}), (m'_{i,j}, \sigma'_{i,j})$ to \mathcal{R} . It stores the tuple $(i, h, pk_{i,h}, \text{msg1}_{i,j}, m'_{i,h}, \sigma'_{i,h})$ in L and continues the simulation. When the reduction outputs a putative solution x' to y , then the meta-reduction also stops with output x' .

Running Time of the Meta-Reduction \mathcal{M} . We first show that the meta-reduction \mathcal{M} has an expected polynomial running time $\text{Time}(\mathcal{M})$, despite the possibly infinitely many loops. This follows by a standard argument.

Let $\epsilon_{i,j}$ denote the conditional probability that we successfully find a valid signature in execution (i, j) and that this is the first successful execution in this row for the transmission $pk_{i,j}, \text{msg1}_{i,j}$ (in any other case the meta-reduction finds a valid entry in the list L and does not enter the loop phase at all). Here, we condition on the

randomness of the reduction and all other fixed message-signature values (such that the probability is only over the choice of $m_{i,j}, r_{i,j}$).

Then, it takes another expected $1/\epsilon_{i,j}$ repetitions to find the second pair, such that for any i, j the expected number of loops (including the main execution and given arbitrary other fixed values) is $1 + \epsilon_{i,j}/\epsilon_{i,j} = 2$. Note that this analysis is under the assumption that we have a restricted resetting scheduling and never run into nested branches. Since each loop thus takes polynomial time on the average only and the simulation of the reduction is polynomially bounded, the claim follows.

Pruning the Meta-Reduction. Recall that our goal is to show that the probability that the reduction \mathcal{R} still succeeds when communicating with \mathcal{M} instead of \mathcal{A} . For an algorithm Z let $\text{Succ}(Z)$ be the event that $V(x', y) = 1$ for $y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^Z(y)$. Then our goal is to show that

$$\text{Prob}[\text{Succ}(\mathcal{M})] := \text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1] \not\approx 0$$

is non-negligible (given $\text{Prob}[\text{Succ}(\mathcal{A})] \not\approx 0$). We now prune the meta-reduction in the simulation above in the sense that in each loop phase our meta-reduction $\mathcal{M}_{r(n)}$ stops after at most $r(n)$ repetitions (and aborts if it has not found a second pair). The polynomial parameter $r(n)$ will be chosen later.

We first analyze the success probability of $\mathcal{M}_{r(n)}$. Clearly,

$$\text{Prob}[\text{Succ}(\mathcal{M})] \geq \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)})]$$

and it therefore suffices to show that the reduction's success probability when interacting with the pruned meta-reduction $\mathcal{M}_{r(n)}$ is non-negligible. Let $\text{Bound}_{r(n)}$ denote the event that in each execution \mathcal{M} rewinds \mathcal{R} at most $r(n)$ times. We then divide the success probability $\text{Prob}[\text{Succ}(\mathcal{M})]$ into the case where \mathcal{M} rewinds the reduction \mathcal{R} more than $r(n)$ times in some loop, and into the other case where \mathcal{M} rewinds the reduction \mathcal{R} at most $r(n)$ times for all loops:

$$\begin{aligned} \text{Prob}[\text{Succ}(\mathcal{M})] &\leq \text{Prob}[\text{Succ}(\mathcal{M}) \wedge \text{Bound}_{r(n)}] + \text{Prob}[\neg \text{Bound}_{r(n)}] \\ &\leq \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \wedge \text{Bound}_{r(n)}] + \text{Prob}[\neg \text{Bound}_{r(n)}] \\ &\leq \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}] + \text{Prob}[\neg \text{Bound}_{r(n)}] \end{aligned}$$

We now define $r(n)$. According to the assumption that the reduction \mathcal{R} with access to

the magic adversary \mathcal{A} succeeds with non-negligible probability, let

$$\text{Prob}[\text{Succ}(\mathcal{A})] \geq \frac{1}{p(n)}$$

for some polynomial $p(n)$ and infinitely many n 's. Let $\mathbb{E}[\text{Time}(\mathcal{M})] = t(n)$ be the (expected) polynomial running time of \mathcal{M} . Now set

$$r(n) := 2 \cdot p(n) \cdot t(n).$$

Using Markov's inequality we can calculate the probability that the event $\neg\text{Bound}_{r(n)}$ happens as

$$\text{Prob}[\neg\text{Bound}_{r(n)}] \leq \text{Prob}[\text{Time}(\mathcal{M}) \geq r(n)] \leq \frac{\mathbb{E}[\text{Time}(\mathcal{M})]}{r(n)} \leq \frac{1}{2p(n)}.$$

Particularly, the probability that \mathcal{M} rewinds the reduction \mathcal{R} more than $r(n)$ times in some execution, is at most $\frac{1}{2p(n)}$.

Comparing the different success probabilities of \mathcal{R} with access to the magic adversary \mathcal{A} and to \mathcal{M} , we have for infinitely many n 's:

$$\begin{aligned} & \text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M})] \\ & \geq \text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}] - \text{Prob}[\neg\text{Bound}_{r(n)}] \\ & \geq \frac{1}{2} \cdot \text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}] \\ & \geq \frac{1}{2} \cdot (\text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]). \end{aligned}$$

In the sequel we assume towards contradiction that $\text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]$ is negligible. We again use transcript-independence of signatures to derive a contradiction.

Description of Adversary $\mathcal{S}_{\text{trans}}^*$. In order to derive a contradiction we build a successful attacker $\mathcal{S}_{\text{trans}}^*$ against the transcript-independence of signatures. This adversary works similar to the previously described adversary $\mathcal{S}_{\text{trans}}^*$ in the proof of Theorem 3.3.7. The difference consists in combining the experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ (where only two executions take place) and the meta-reduction (where many interactions take place). To overcome the difference the adversary picks a random subroutine call k among all at most q^2 ones and tries to insert the data provided by its experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ in one of the at most $r(n)$ repetitions, which $\mathcal{M}_{r(n)}$ makes to find the second message-signature pair.

More formally, in a first step the adversary $\mathcal{S}_{\text{trans}}^*$ computes an instance $y \leftarrow I(1^n)$ of

a cryptographic problem P and selects a random index $k \in \{1, \dots, q^2\}$. It proceeds with the black-box simulation of the reduction \mathcal{R} which takes the instance y as input. During the simulation of \mathcal{R} adversary $\mathcal{S}_{\text{trans}}^*$ maintains a copy of $\mathcal{M}_{r(n)}$ and mainly uses this algorithm to compute the answers.

Only in the first k subroutine calls of $\mathcal{M}_{r(n)}$ (in which the meta-reduction loops to compute the second pair) algorithm $\mathcal{S}_{\text{trans}}^*$ diverges from $\mathcal{M}_{r(n)}$'s strategy as follows. For the first $k - 1$ of the runs in which $\mathcal{M}_{r(n)}$ branches into the extraction procedure for execution (i, j) , adversary $\mathcal{S}_{\text{trans}}^*$ uses its oracle Σ to compute a signature $\sigma_{i,j}^*$ for an independent random message $m_{i,j}^*$. It stores $(i, j, pk_{i,j}, \text{msg1}_{i,j}, m_{i,j}^*, \sigma_{i,j}^*)$ in L and uses this message-signature pair instead and ignores the meta-reduction's pair (if it finds one). Another exception is the way the answers for the k -th subprogram execution are derived. Here the adversary $\mathcal{S}_{\text{trans}}^*$ behaves as follows. Let $(pk_{i,j}, \text{msg1}_{i,j})$ be the data initially sent by the reduction in this execution. Adversary $\mathcal{S}_{\text{trans}}^*$ executes experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ using \mathcal{R} , i.e., $\mathcal{S}_{\text{trans}}^*$ uses the public key sent by \mathcal{R} as his public key during the experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$. It selects two random messages m'_{-1}, m'_0 as the challenges and outputs (pk, m'_{-1}, m'_0) . The adversary $\mathcal{S}_{\text{trans}}^*$ relays the first instances of the signature issuing protocol between \mathcal{R} and \mathcal{U} and checks whether \mathcal{U} is able to derive a signature. If the signature-derivation check returns $c = 0$, i.e., indicating that the user should not be able to generate a valid signature, then $\mathcal{S}_{\text{trans}}^*$ stops, outputting a random bit b^* . Otherwise, $\mathcal{S}_{\text{trans}}^*$ proceeds with the simulation as follows. For the other signature generation for m'_0 , adversary $\mathcal{S}_{\text{trans}}^*$ guesses how many rewindings (of \mathcal{R}) are necessary in order to derive another pair. For this, it selects a random index $t \in \{1, \dots, r(n)\}$ and computes $t - 1$ random messages m^ℓ as well as $t - 1$ random strings r^ℓ for $\ell = 1, 2, \dots, t - 1$. During the ℓ -th repetition for $\ell < t$, adversary $\mathcal{S}_{\text{trans}}^*$ executes an instance of the user algorithm \mathcal{U} using the coins r^ℓ as well as the message m^ℓ . If one of these $t - 1$ instances already yields a valid signature, then $\mathcal{S}_{\text{trans}}^*$ aborts and outputs a random bit b^* as its final output.

Otherwise, at the beginning of the t -th rewinding (in which $\mathcal{S}_{\text{trans}}^*$ expects to generate a signature successfully), the adversary forwards $\text{msg1}_{i,j}$ to the external user instance (holding key $pk_{i,j}$ and message m'_i) in experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$ to receive an answer msg2 . The meta-reduction uses this answer in all executions in this row i with first transmission $pk_{i,j}, \text{msg1}_{i,j}$. Additionally, in all these executions (except for (i, j)) the adversary runs the signature-derivation checks to see if an earlier execution would yield a valid signature. If any of these checks returns $c = 1$, i.e., that the user should be able to generate a valid signature, the $\mathcal{S}_{\text{trans}}^*$ immediately stops with a random output bit b^* . In any other case, the adversary returns \perp to the reduction as the reply to msg3 .

For execution (i, j) the adversary takes the reduction's answer msg3 and forwards it to the external user instance. If the interaction with the external user instance does

not yield a valid signature (or, more generally, if both σ'_{-1}, σ'_0 are invalid), then $\mathcal{S}_{\text{trans}}^*$ stops outputting a random bit b^* . Otherwise, it returns to \mathcal{R} the message-signature pairs $(m_{i,j}, \sigma_{i,j}, m'_1, \sigma'_1)$, where $m_{i,j}, \sigma_{i,j}$ have been generated during the first execution and m'_0, σ'_0 has been derived either with the help of Σ or through the interaction with \mathcal{U} in experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$. We remark that, for each valid execution, $\mathcal{S}_{\text{trans}}^*$ also enters the rewind phases and seeks a second message-signature pair in $r(n)$ loops. If the meta-reduction fails to find such a pair in any of these rewinding steps then we let $\mathcal{S}_{\text{trans}}^*$ immediately abort, outputting a random bit b^* . If no premature abort happens then, eventually, the reduction outputs a putative solution x' for a cryptographic problem P . The final output of $\mathcal{S}_{\text{trans}}^*$ (if it has not aborted before) is $V(x', y) \oplus 1$.

Analysis of Adversary $\mathcal{S}_{\text{trans}}^*$. To analyze the success probability of $\mathcal{S}_{\text{trans}}^*$ we define the following hybrid oracles. Consider a run of the reduction with q^2 oracles, but where we use the strategy of the meta-reduction $\mathcal{M}_{r(n)}$ in the all executions, except that we use the magic adversary in the first k subprocedure executions to replace the second pair (or to find a pair at all if $\mathcal{M}_{r(n)}$ has not found one). We denote this “oracle matrix” by \mathcal{H}_k . Accordingly, we write $\text{Succ}(\mathcal{H}_k)$ for the event that the reduction \mathcal{R} successfully outputs a solution x' to $y \leftarrow I(1^n)$ when interacting with such a hybrid oracle set.

By construction we have identical behavior for the extreme hybrids to the adversary’s attack and the execution of the meta-reduction, respectively, where we use in the former case the fact that \mathcal{A} computes the pairs (m, r) with the help of the q -wise independent hash function (just as $\mathcal{M}_{r(n)}$ picks fresh random values):

$$\text{Prob}[\text{Succ}(\mathcal{H}_{q^2})] = \text{Prob}[\text{Succ}(\mathcal{A})] \quad \text{and} \quad \text{Prob}[\text{Succ}(\mathcal{H}_0)] = \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)})].$$

We will now set this in relationship to the success probability of $\mathcal{S}_{\text{trans}}^*$ predicting b with its output b^* .

First, we collect the cases that $\mathcal{S}_{\text{trans}}^*$ aborts prematurely, returning a random bit b^* . This happens if event $\neg\text{Bound}_{r(n)}$ occurs, if the adversary’s guess t for the right repetition has been wrong (event $\neg\text{Guess}$), or if the guess has been right but the signature-derivation check returns a wrong answer, saying that the user was able to compute a signature while he was not (event $\neg\text{SDCh}^+$). Similarly, the simulation may be erroneous if the check returns that the user is not able to derive a signature but he actually is (event $\neg\text{SDCh}^-$).

It is easy to see that the probability for event $\neg\text{SDCh}^+ \vee \neg\text{SDCh}^-$ is negligible by the signature-derivation check; else one can easily build a successful attacker from $\mathcal{S}_{\text{trans}}^*$ against this property. Hence, this simulation error can only affect the adversary’s success probability for predicting b negligibly. From now on we therefore implicitly

condition on the event $\text{SDCh}^+ \wedge \text{SDCh}^-$ that all signature-derivation checks return the right answer.

For the probability of predicting b we now take into account the cases that events $\text{Bound}_{r(n)}$ and Guess do not hold (in which case $\mathcal{S}_{\text{trans}}^*$ returns a random bit b^* and succeeds with probability $\frac{1}{2}$), and derive:

$$\begin{aligned}
& \text{Prob}[b = b^*] \\
&= \text{Prob}[b = b^* \mid \neg \text{Bound}_{r(n)} \vee \neg \text{Guess}] \cdot \text{Prob}[\neg \text{Bound}_{r(n)} \vee \neg \text{Guess}] \\
&\quad + \text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}] \cdot \text{Prob}[\text{Bound}_{r(n)} \wedge \text{Guess}] \\
&= \frac{1}{2} \cdot (1 - \text{Prob}[\text{Bound}_{r(n)} \wedge \text{Guess}]) \\
&\quad + \text{Prob}[\text{Bound}_{r(n)} \wedge \text{Guess}] \cdot \text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}] \\
&= \frac{1}{2} + \text{Prob}[\text{Bound}_{r(n)} \wedge \text{Guess}] \cdot (\text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}] - \frac{1}{2})
\end{aligned}$$

Next, note that $\text{Prob}[\text{Guess} \mid \text{Bound}_{r(n)}] = 1/r(n)$, because given that we always find another message-signature pair in $r(n)$ loops we pick the right one to insert the data with this probability. Since we also have $\text{Prob}[\text{Bound}_{r(n)}] \geq 1 - 1/2p(n)$ we conclude that $\text{Prob}[\text{Bound}_{r(n)} \wedge \text{Guess}]$ is non-negligible. But then it suffices to show that the probability of predicting b under these two conditions is bounded away from $\frac{1}{2}$ by a non-negligible amount. This follows by refining the view with respect to bit b :

$$\begin{aligned}
& \text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}] - \frac{1}{2} \\
&= \text{Prob}[b = 1] \cdot \text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}, b = 1] \\
&\quad + \text{Prob}[b = 0] \cdot \text{Prob}[b = b^* \mid \text{Bound}_{r(n)}, \text{Guess}, b = 0] - \frac{1}{2} \\
&= \frac{1}{2} \cdot (1 - \text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 1]) \\
&\quad + \frac{1}{2} \cdot \text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 0] - \frac{1}{2} \\
&= \frac{1}{2} \cdot (\text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 1] \\
&\quad - \text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 0])
\end{aligned}$$

Taking the random choice k of $\mathcal{S}_{\text{trans}}^*$ into account we obtain:

$$\begin{aligned}
& \text{Prob}[b^* = 0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 1] - \text{Prob}[b^* = 0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 0] \\
&= \sum_{k_0=1}^{q^2} (\text{Prob}[b^* = 0 \wedge k = k_0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 1] \\
&\quad - \text{Prob}[b^* = 0 \wedge k = k_0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 0])
\end{aligned}$$

$$\begin{aligned}
&= \frac{1}{q^2} \cdot \sum_{k_0=1}^{q^2} (\text{Prob}[b^* = 0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 1, k = k_0]) \\
&\quad - \text{Prob}[b^* = 0 \mid \text{Guess}, \text{Bound}_{r(n)}, b = 0, k = k_0]
\end{aligned}$$

The probability that $\mathcal{S}_{\text{trans}}^*$ outputs $b^* = 0$, given that the guess is right, the number of repetitions is bounded and $b = 1$ (i.e., $\mathcal{S}_{\text{trans}}^*$ forwards the signature generated by Σ) and that $k = k_0$, equals the probability for $\text{Succ}(\mathcal{H}_{k_0})$ (under the condition $\text{Bound}_{r(n)}$). Similarly, under these conditions and that $b = 0$, i.e., that $\mathcal{S}_{\text{trans}}^*$ inserts the communication with the user from experiment $\text{trans-ind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$, the probability for $b^* = 0$ is identical to the one for $\text{Succ}(\mathcal{H}_{k_0-1})$ (under the condition $\text{Bound}_{r(n)}$). The latter also relies on Corollary 3.3.5 (page 22) that Σ succeeds in producing a signature with overwhelming probability if the first execution is valid, because it is guaranteed that $\mathcal{S}_{\text{trans}}^*$ obtains the two signatures from the user instances in the experiment (if Σ would fail then the transcript-independence experiment would return \perp for all three executions). We ignore this negligible error in Corollary 3.3.5 for simplicity and conclude that

$$\begin{aligned}
&\text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 1] - \text{Prob}[b^* = 0 \mid \text{Bound}_{r(n)}, \text{Guess}, b = 0] \\
&= \frac{1}{q^2} \cdot \sum_{k_0=1}^{q^2} (\text{Prob}[\text{Succ}(\mathcal{H}_{k_0}) \mid \text{Bound}_{r(n)}] - \text{Prob}[\text{Succ}(\mathcal{H}_{k_0-1}) \mid \text{Bound}_{r(n)}]) \\
&= \frac{1}{q^2} \cdot (\text{Prob}[\text{Succ}(\mathcal{H}_{q^2}) \mid \text{Bound}_{r(n)}] - \text{Prob}[\text{Succ}(\mathcal{H}_0) \mid \text{Bound}_{r(n)}]) \\
&= \frac{1}{q^2} \cdot (\text{Prob}[\text{Succ}(\mathcal{H}_{q^2})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]) \\
&= \frac{1}{q^2} \cdot (\text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]).
\end{aligned}$$

where we used the fact that the success probability for the case \mathcal{H}_{q^2} is independent of event $\text{Bound}_{r(n)}$ (because in this experiment all second message-signature pairs are provided by oracle Σ , independently of whether the pruned meta-reduction finds a second pair). Plugging this latter term into the previous equation for $\text{Prob}[b = b^*]$, we obtain

$$\begin{aligned}
&\text{Prob}[b = b^*] \\
&\geq \frac{1}{2} + \frac{1}{2r(n)q^2(n)} \cdot \left(1 - \frac{1}{2p(n)}\right) \cdot (\text{Prob}[\text{Succ}(\mathcal{A})] - \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]).
\end{aligned}$$

By assumption, the probability for a success of \mathcal{R} with the magic adversary is non-negligible, whereas for the pruned meta-reduction (under condition $\text{Bound}_{r(n)}$) it drops to negligible. But then we have derived a successful attacker against the transcript independence of signatures. It follows that our assumption $\text{Prob}[\text{Succ}(\mathcal{M}_{r(n)}) \mid \text{Bound}_{r(n)}]$

being negligible must have been wrong. Since

$$\begin{aligned} \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)})] &\geq \text{Prob}[\text{Bound}_{r(n)}] \cdot \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)} \mid \text{Bound}_{r(n)})] \\ &\geq \left(1 - \frac{1}{2^{p(n)}}\right) \cdot \text{Prob}[\text{Succ}(\mathcal{M}_{r(n)} \mid \text{Bound}_{r(n)})] \end{aligned}$$

it follows that the success probability of $\mathcal{M}_{r(n)}$ must be non-negligible, too. Also note that $\mathcal{M}_{r(n)}$ gives us a solver running in fixed polynomial time. \square

Let us re-capture the step where we used the fact that our scheme has three moves only. For this we look at the construction of $\mathcal{S}_{\text{trans}}^*$, showing that any significant difference in the reduction's success probability when communicating with \mathcal{A} or with $\mathcal{M}_{r(n)}$ can be used to break transcript-independence of signatures. This adversary uses the external procedures Σ and \mathcal{U} to derive the second pair (in one of the $r(n)$ repetitions). In particular, the external user algorithm cannot be reset according to the transcript-independence experiment.

Fortunately, since the blind signature scheme has only three moves we can simply insert the same second message of the external user in all executions with the same $pk, \text{msg}1$. In other words, resets are easy to simulate. If the blind signature scheme had four or more moves, however, the reduction could reset each execution at different points, possibly extracting some knowledge about the message and/or the randomness of the user. Adversary $\mathcal{S}_{\text{trans}}^*$ could in general not simulate these steps without resetting the external user algorithm.

3.5 Impossibility Result for Computationally Blind Signature Schemes

Here we extend our result to computationally blind signature schemes.

3.5.1 Preliminaries

We augment the definition of blindness by allowing the malicious signer \mathcal{S}^* to invoke an oracle Σ . As before, we note that Σ will break unforgeability and the definition below says that blindness should still hold, even if one can forge signatures. As an example, consider Chaum's scheme, where perfect blindness is preserved even if one can break RSA.

Definition 3.5.1 (Blind Signature Scheme Relative to an Oracle Σ). A secure blind signature scheme $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ is called computationally blind relative to an oracle Σ if, for any efficient algorithm \mathcal{S}^* working in modes *find*, *issue* and *guess* the probability that the following experiment $\text{Blind}_{\mathcal{S}^*}^{\text{BS}}$ evaluates to 1 is negligibly close to $1/2$, where

Experiment $\text{Blind}_{\mathcal{S}^*, \Sigma}^{\text{BS}}(n)$
 $(pk, m_0, m_1, \text{st}_{\text{find}}) \leftarrow \mathcal{S}^{*, \Sigma}(\text{find}, 1^n)$
 $b \leftarrow \{0, 1\}$
 $\text{st}_{\text{issue}} \leftarrow \mathcal{S}^{*, \Sigma}(\langle \cdot, \mathcal{U}(pk, m_b) \rangle^1, \langle \cdot, \mathcal{U}(pk, m_{1-b}) \rangle^1, \Sigma)(\text{issue}, \text{st}_{\text{find}})$
and let σ_b, σ_{1-b} denote the (possibly undefined) local outputs of $\mathcal{U}(pk, m_b)$ resp. $\mathcal{U}(pk, m_{1-b})$.
set $(\sigma_0, \sigma_1) = (\perp, \perp)$ if $\sigma_0 = \perp$ or $\sigma_1 = \perp$
 $b^* \leftarrow \mathcal{S}^{*, \Sigma}(\text{guess}, \sigma_0, \sigma_1, \text{st}_{\text{issue}})$
return 1 iff $b = b^*$.

Key-Validity Checks. For our impossibility result we need an additional requirement on the blind signature scheme which allows to check publicly whether a maliciously chosen public key has a matching secret key (we call this a key-validity check). We need this property because our result is based on a (different) generic forgery oracle Σ . In the statistical case the forgery oracle has basically searched for a collision for the transcript, but in the computational case such collisions may not even exist. Hence, instead we let Σ now compute a secret key from the public key and then run an execution between the honest user and the honest signer for this secret key. The key-validity check tells us whether this strategy succeeds or not.

Definition 3.5.2 (Key-Validity Check). A blind signature scheme BS allows (computational resp. statistical) key-validity checks if there exists an efficient algorithm KVCh such that for any (efficient resp. unbounded) algorithm \mathcal{S}^* working in modes *find* and *issue* the probability that the following experiment $\text{KeyValCheck}_{\mathcal{S}^*, \text{KVCh}}^{\text{BS}}$ evaluates to 1 is negligible, where

Experiment $\text{KeyValCheck}_{\mathcal{S}^*, \text{KVCh}}^{\text{BS}}$
 $(pk, m, \text{st}) \leftarrow \mathcal{S}^*(\text{find}, 1^n)$
 $c \leftarrow \text{KVCh}(1^n, pk)$
return 1 if
 $c = 1$ but there does not exist sk' with $(sk', pk) \in [\text{KG}(1^n)]$, or
 $c = 0$ but there exists sk' with $(sk', pk) \in [\text{KG}(1^n)]$.

If the above holds even if \mathcal{S}^* gets access to an oracle Σ then we say that the scheme has (computational resp. statistical) key-validity checks relative to Σ .

The schemes of Pointcheval-Stern and of Boldyreva, for example, allow to implement such a key-validity check by verifying that the discrete-log groups are admissible (e.g., prime order sub group) and that the values are proper group elements.

Generic Forgery Oracle. To define our generic forgery oracle Σ^c allowing \mathcal{A} to break unforgeability we first outline the idea for the case of Chaum’s blind signature scheme. Namely, assume that the RSA-exponent e in Chaum’s scheme has a unique matching secret exponent d . Algorithm $\Sigma(pk, m)$ then computes the inverse exponent d to the RSA key (N, e) and sets $\sigma = H(m)^d \bmod N$ for the hash function description H in the public key. Note that the message deterministically identifies the signature, and the distribution of Σ ’s output is therefore identical to the one of an honest user.

The above example can be generalized to any blind signature scheme and the following generic forgery oracle (which only depends on the blind signature scheme in question):

Definition 3.5.3 (Generic Forgery Oracle). *For a blind signature scheme BS the generic forgery oracle $\Sigma^c = (\Sigma_{sk}^c, \Sigma_{ex}^c)$ consists of two algorithms, where*

SIGNING KEY GENERATION. *Algorithm Σ_{sk}^c on input (pk, m) enumerates all possible random inputs to KG which lead KG for input 1^n to produce pk . The oracle uniformly picks one of those random strings and returns the corresponding secret key sk_{Σ^c} which KG outputs for input 1^n and for this string. If no such string exists, then Σ_{sk}^c returns \perp .*

EXECUTION. *Algorithm Σ_{ex}^c takes as input pk, m and a key sk_{Σ^c} and runs an execution between $\mathcal{S}(sk_{\Sigma^c})$ and an instance of the honest user $\mathcal{U}(pk, m)$. This eventually yields a signature σ (possibly $\sigma = \perp$) output by the user, and Σ_{ex}^c then returns σ .*

We note that any algorithm with oracle access to Σ^c can call each suboracle individually. Vice versa, when calling Σ^c with (pk, m) we assume that Σ^c internally first executes $\Sigma_{sk}^c(pk, m)$ to derive sk_{Σ^c} and then returns Σ_{ex}^c ’s answer for input pk, m, sk_{Σ^c} . Additionally, by the completeness of the blind signature scheme the forgery oracle always returns a valid signature when called for a public key generated by the honest signer.

Transcript Independence. We briefly discuss that transcript-independence in the computational case (for our generic forgery oracle here) holds because of the blindness relative to Σ^c . We remark that, since Σ^c does not depend on the transcript at all, the prerequisites do not include signature derivation checks:

Proposition 3.5.4. *Every blind signature scheme, which is blind relative to the generic forgery oracle Σ^c , also has computational transcript-independent signatures (with respect to Σ^c).*

Proof. The proof carries over from the statistical case (Proposition 3.3.4) with slight changes. So assume that there exists a signer $\mathcal{S}_{\text{trans}}^*$ in experiment $\text{transc-ind}_{\mathcal{S}_{\text{trans}}^*, \Sigma}^{\text{BS}}(n)$ with the generic forgery oracle Σ^c which outputs $b^* = b$ with non-negligible probability beyond $1/2$. Then we construct an adversarial controlled signer $\mathcal{S}_{\text{blind}}^*$ against the blindness (with oracle access to Σ^c) as follows. Algorithm $\mathcal{S}_{\text{blind}}^*$ invokes $\mathcal{S}_{\text{trans}}^*(\text{init}, 1^n)$ to get $(pk, \text{st}_1, m_{-1}, m_0)$ and also runs $\Sigma_{\text{sk}}^c(pk, m_0)$ to get sk_{Σ^c} . It outputs $(pk, (\text{st}_1, sk_{\Sigma^c}), m_0, m_0)$ as the initial output in the blindness experiment.

In the following $\mathcal{S}_{\text{blind}}^*$ impersonates the honest user \mathcal{U} for input (pk, m_{-1}) in the left user instance of $\mathcal{S}_{\text{trans}}^*$ by following the user algorithm. In the right user instance for the transcript-independence adversary $\mathcal{S}_{\text{blind}}^*$ relays all the communication with its first external user instance (for input (pk, m_0)). It also invokes the second user instance (also for (pk, m_0)) and uses algorithm $\mathcal{S}(sk_{\Sigma^c})$ for key sk_{Σ^c} to answer the user.

Algorithm $\mathcal{S}_{\text{blind}}^*$ eventually obtains $(m_0, \sigma_0, m_0, \sigma_1)$ (without knowing if σ_0 origins from the communication between $\mathcal{S}_{\text{trans}}^*$ and the user, or from the internally simulated algorithm $\mathcal{S}(sk_{\Sigma^c})$ and the user). If $\sigma_0 = \perp$ then it also sets $\sigma_{-1} \leftarrow \perp$. In any case it forwards $(m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ to $\mathcal{S}_{\text{trans}}^*$ and returns this algorithm's output b^* as its decisional bit.

For the analysis observe that for $b = 0$ in the blindness experiment the data provided to $\mathcal{S}_{\text{trans}}^*$ corresponds exactly to the case $b = 0$ there. Also, the case $b = 1$ in the blindness experiment is exactly like the case $b = 1$ in the transcript-independence experiment, because the generic forgery oracle also runs an instance between $\mathcal{S}(sk_{\Sigma^c})$ and $\mathcal{U}(pk, m_0)$. This implies that if $\mathcal{S}_{\text{trans}}^*$ succeeds with non-negligible probability over $1/2$, then $\mathcal{S}_{\text{blind}}^*$ also succeeds with non-negligible probability bounded away from $1/2$. \square

Pseudorandom Functions. In order to prove our impossibility result, we take advantage of pseudorandom functions, similar to our deployment of q -wise independent hash functions. To this end we define pseudorandom functions in the presence of an oracle Σ^c and magic adversaries with access to Σ^c . In the following let $\rho(n)$ be the length of the randomness used by an honest user for an execution of the signing protocol. As we will later view Σ^c to be an integral part of the magic adversary and thus let the adversary provide the randomness $s \in \{0, 1\}^{\psi(n)}$ required by oracle Σ^c , we also grant the distinguisher in the pseudorandom experiment here access to the augmented (deterministic) oracle $\Sigma^{c, \text{aug}}$ which now takes pk, m and randomness s as input and returns σ :

Definition 3.5.5 (Pseudorandom Function Relative to Oracle). Let \mathcal{R}_n be the set of all functions $f : \{0, 1\}^* \rightarrow \{0, 1\}^{2n+\rho(n)+\psi(n)}$, Σ be an oracle and PRF be an algorithm which takes as input $k \in \{0, 1\}^n$ and $x \in \{0, 1\}^*$ and returns a value of length $2n + \rho(n) + \psi(n)$. Then PRF is called a pseudorandom function relative to oracle Σ^{aug} if for every efficient algorithm D the following holds:

$$\text{Prob} [D^{\Sigma^{\text{aug}}, \text{PRF}(k, \cdot)}(1^n) = 1] - \text{Prob} [D^{\Sigma^{\text{aug}}, f(\cdot)}(1^n) = 1] \approx 0,$$

where the probability in the first case is taken over the internal coin tosses of D and over the choice of $k \leftarrow \{0, 1\}^n$, and in the second case over the internal coin tosses of D and over the choice of $f \leftarrow \mathcal{R}_n$.

An equivalent way of defining pseudorandom functions (relative to oracles) is to give the distinguisher (in addition to Σ^{aug}) either access to q functions $\text{PRF}(k_1, \cdot), \dots, \text{PRF}(k_q, \cdot)$ for independent keys k_1, \dots, k_q , or to q independent random functions f_1, \dots, f_q . A standard hybrid argument shows that for polynomial $q = q(n)$ a function is pseudorandom according to this definition if and only if it is pseudorandom according to Definition 3.5.5 above (even in presence of Σ^{aug}). Below we will make use of this version with multiple oracles.

The New Magic Adversary. Given the pseudorandom function relative to an oracle we augment our “magic” adversary through access to a pseudorandom function PRF. Informally, the adversary again runs the issuing protocol with the signer in the role of the honest user once. However, it now generates the message (and the user’s randomness) as the result of the pseudorandom function to the public key and the first message of the signer. Again, in the case that the single execution yields a valid signature, then the magic adversary here also creates another valid signature via $\Sigma^{c, \text{aug}}$. Since we view the oracle Σ^c as a subroutine of the magic adversary the randomness for Σ^c is now also provided explicitly by the adversary and derived through the pseudorandom function (we note that the length $\psi(n)$ of this randomness is only polynomial by construction of the generic forgery oracle):

Definition 3.5.6 (Magic Adversary with Access to PRF, Σ^c). The magic adversary \mathcal{A}_{PRF} for input pk and access to the generic forgery oracle $\Sigma^{c, \text{aug}}$ and communicating with an oracle $\langle \mathcal{S}(sk), \cdot \rangle^1$ works as described in the following steps:

select a key k for the pseudorandom function PRF
run an execution $\langle \mathcal{S}(sk), \mathcal{U}(pk, m'_0; r'_0) \rangle$ in the role of an honest user, where
 $(m'_0, m'_1, r'_0, s'_0) \leftarrow \text{PRF}(k, pk || \text{msg}1)$ is generated as the result of the
pseudorandom function applied to the public key pk and the first

message msg_1 of \mathcal{S} ; let σ'_0 denote the resulting signature.
 if $\forall f(pk, m'_0, \sigma'_0) = 1$ then let $\sigma'_1 \leftarrow \Sigma^{c, \text{aug}}(pk, m'_1; s'_0)$ else set $\sigma'_0 \leftarrow \perp, \sigma'_1 \leftarrow \perp$
 return $(m'_0, \sigma'_0, m'_1, \sigma'_1)$

It follows again from the completeness of BS together with the construction of the generic forgery oracle (which works even for pseudorandom input instead of truly random coins) that the magic adversary succeeds in the unforgeability experiment with probability negligibly close to 1.

We note that, if we only consider reductions with an a-priori fixed number q of resets in each row, then we could let the adversary use its randomness to implement q -wise independent hash functions instead of pseudorandom functions (similar to [BL02] and our result for statistical blindness). However, in case of computationally (but not statistically) blind signature schemes relative to $\Sigma^{c, \text{aug}}$, pseudorandom functions relative to $\Sigma^{c, \text{aug}}$ exist anyway and therefore do not require an additional assumption. This follows as we then have one-way functions relative to $\Sigma^{c, \text{aug}}$ [Gol90] and can apply the (relativizing) constructions [GGM86, HILL99a] to derive pseudorandom functions relative to $\Sigma^{c, \text{aug}}$.

3.5.2 Impossibility Result

The following theorem extends our impossibility result to the case of computational blind signature schemes.

Theorem 3.5.7. *Let BS be a three-move blind signature scheme, which is blind relative to the generic forgery oracle Σ^c and which has (computational) signature-derivation checks and (computational) key-validity checks relative to Σ^c . Let PRF be a pseudorandom function relative to $\Sigma^{c, \text{aug}}$. Then there is no resetting (with restricted cross-resets) black-box reduction from unforgeability of the blind signature scheme BS to a hard non-interactive problem.*

Note again that such pseudorandom functions exist if the blind signature scheme is computationally but not statistically blind.

The high-level idea of the proof of Theorem 3.5.7 is similar to the proof of Theorem 3.4.4 with the difference that we investigate computational blind signature schemes and that the reduction is allowed to reset the adversary as often as required (and not a fixed number). In order to handle resetting attacks we divide the proof in two parts. In the first part we modify the magic adversary \mathcal{A} by substituting the pseudorandom function through a random function and show that this difference does not change the success probability of the reduction non-negligibly. This makes the resetting executions

essentially independent and facilitates the simulation of the reduction through the meta-reduction.

In the main step of the proof, we then construct a meta-reduction \mathcal{M} which mimics the adversarial behavior (without the help of Σ^c) by rewinding the reduction \mathcal{R} as described in the proof of Theorem 3.4.4. The difference of both constructions consists in the last step of the meta-reduction. When \mathcal{M} has obtained two message-signature pairs, then \mathcal{M} runs the key-validity check. In the case that this test evaluates to 0, i.e., that Σ^c should not be able to output a corresponding secret key and the magic adversary thus fails to produce a forgery, then \mathcal{M} responds with $(\sigma_0, \sigma_1) = (\perp, \perp)$. Otherwise, if the test outputs 1, then \mathcal{M} forwards both signatures to \mathcal{R} . We again show that \mathcal{M} 's behavior and the one of the magic adversary are indistinguishable to \mathcal{R} by the transcript independence of signatures.

Proof. The proof consists of the following steps. We first show that we can safely replace the pseudorandom function used by \mathcal{A} through a truly random function. Then, we describe the meta-reduction \mathcal{M} with expected polynomial running time. To ensure fixed polynomial running time we next prune the meta-reduction to $\mathcal{M}_{r(n)}$. We prove that the success probability of this pruned meta-reduction is close to the one of the reduction communicating with \mathcal{A} , yielding our desired efficient solver for the underlying problem.

Replacing PRF by Random Functions. Let \mathcal{A}_{PRF} be the magic adversary with access to the pseudorandom function PRF and to the generic forgery oracle $\Sigma^{\text{c, aug}}$. We first modify \mathcal{A}_{PRF} to \mathcal{A}_f by replacing the pseudorandom function through a random function f in each row, chosen at random when initialized. In particular, different copies with the same random string rely on the same random function. We argue that this does not make a non-negligible difference for the reduction \mathcal{R} .

Assume towards contradiction that the reduction \mathcal{R} outputs a valid solution x' with non-negligible probability if \mathcal{R} receives message-signature pairs $(m_0, \sigma_0), (m_1, \sigma_1)$ from the magic adversary \mathcal{A}_{PRF} (i.e., with access to PRF), but succeeds only with negligible probability if the message-signature pairs are generated by the magic adversary \mathcal{A}_f (i.e., with access to random functions). We then construct a distinguisher D who exploits this difference in these probabilities to successfully distinguish functions from PRF and random functions. Here we use the version with multiple independent oracles, discussed after Definition 3.5.5.

The distinguisher D has access to $\Sigma^{\text{c, aug}}$ and to q function oracles $\mathcal{F}_1, \dots, \mathcal{F}_q$ which accept binary strings as input and return strings of length $2n + \rho(n) + \psi(n)$. The function oracles either compute independent pseudorandom functions $\text{PRF}(k_i, \cdot)$ or truly random

functions f_1, \dots, f_q . The distinguisher works as follows. It first generates an instance $y \leftarrow I(1^n)$ of a cryptographic problem P . It starts to simulate the reduction \mathcal{R} on input y , simulating the adversary copies in row i as described in Construction 3.4.3, but using the function oracle \mathcal{F}_i instead of the pseudorandom function. When the reduction finally outputs an alleged solution x' the distinguisher D returns $b' \leftarrow V(x', y)$.

According to our assumption that the reduction \mathcal{R} only succeeds with negligible probability if the message-signatures pairs are generated by \mathcal{A}_f (i.e., by the magic adversary \mathcal{A} with access to truly random functions) we have

$$\text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}_f}(y) : V(x', y) = 1 \mid \mathcal{A}_f] \approx 0.$$

By construction this is identical to the probability that the distinguisher D returns 1, given that the function oracles $\mathcal{F}_1, \dots, \mathcal{F}_q$ are given by random functions. On the other hand,

$$\text{Prob}[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}_{\text{PRF}}}(y) : V(x', y) = 1 \mid \mathcal{A}_{\text{PRF}}] \not\approx 0.$$

and this probability equals the probability that D outputs 1 if the function oracles implement pseudorandom functions. Overall,

$$\text{Prob}[D^{\Sigma^{\text{c, aug}}, \text{PRF}(k_1, \cdot), \dots, \text{PRF}(k_q, \cdot)}(1^n) = 1] - \text{Prob}[D^{\Sigma^{\text{c, aug}}, f_1, \dots, f_q}(1^n) = 1] \not\approx 0.$$

But this contradicts the pseudorandomness of PRF.

In conclusion, the magic adversary with access to the truly random function now can be viewed as follows. Each time the reduction sends a new pair $(pk, \text{msg1})$ in a row the adversary essentially creates the message-signature pairs independently. For different rows this even holds for the same $pk, \text{msg1}$ as the random function is independent from the ones for the other rows. We can now also switch from $\Sigma^{\text{c, aug}}$ to Σ^{c} as long as we guarantee that Σ^{c} gives identical answers for executions with the same $pk, \text{msg1}$; this can be easily implemented by table look-ups.

The Final Step. Given that we can again assume independent random choices for the adversary we next discuss the necessary changes to make the proof of Theorem 3.4.4 go through in this case.

The meta-reduction \mathcal{M} here essentially behaves as the one in the proof of Theorem 3.4.4, but differs in the last step for each loop phase. Namely, after \mathcal{M} has computed two message-signature pairs, it runs the key-validity check on the corresponding public key. If this check outputs 0, then \mathcal{M} sets $\sigma'_0, \sigma'_1 \leftarrow \perp$. Otherwise, it forwards both signatures to \mathcal{R} .

The analysis of the meta-reduction is almost identical to the analysis of the meta-reduction in the proof of Theorem 3.4.4. We again show that any noticeable difference for the reduction when communicating with the magic adversary or with the meta-reduction yields a contradiction to the transcript-independence (but now for the computational case). The only difference is that, in that proof we referred to Corollary 3.3.5 to ensure that a failing Σ^c in the transcript-independence experiment does not prevent the adversary $\mathcal{S}_{\text{trans}}^*$ from obtaining the two signatures the meta-reduction would derive. Here, running the key-validity check by the meta-reduction provides the same guarantee. Only this time we let the meta-reduction artificially fail then, and the conclusion therefore remains true. \square

3.5.3 Extension to Four-Move Blind Signature Schemes

A natural question is if our technique can also be used to rule out four-move blind signature schemes. This, however, does not seem to be the case. First observe that in a four-move signature-issue protocol the user algorithm sends the first message to the signer. The problem is that the user might commit itself to the message in this move. In this case, the reduction might extract the message by resetting the user algorithm. Applying our technique might allow the meta-reduction to derive further signatures, but only on the same message. From a more abstract point of view, the reduction might reset the user (and therefore the meta-reduction) to the point where it has sent the first message and request another response from the user instance. Our technique then breaks down, because when we show indistinguishability of the meta-reduction's strategy and the one of the magic adversary via blindness, we plug in the communication from an external *non-resettable* execution into the reduction at one step.

4 Blind Signatures and Their Applications to Adaptive OT

4.1 Introduction

The security requirements for blind signature schemes have been formalized by Juels et al. [JLO97a] and by Pointcheval and Stern [PS00a]. Although these widely used definitions give basic security guarantees, blindness only holds in a restricted sense when it comes to aborted executions. That is, prior work does not guarantee blindness in case the signer is able to learn which of two executions aborted (even if one execution aborts only after the protocol has concluded). However, in e-cash scenarios an honest user, unable to eventually derive a valid coin, will most likely complain to the malicious bank afterwards. From a theoretical point of view this property is also desirable as it is needed (among other properties) to build adaptive oblivious transfer from any unique blind signature scheme in the random oracle. Such a transformation has been suggested by Camenisch et al. [CNS07] who consider a stronger kind of aborts where a cheating signer may be able to make the user algorithm fail depending on the message being signed,¹ and where the malicious signer is informed afterwards which execution has failed (if any).

Related Work. As mentioned before, Camenisch et al. [CNS07] have already considered the limitations of the standard blindness notion. They have introduced an extension called *selective-failure blindness* in which the malicious signer should not be able to force an honest user to abort the signature issue protocol because of a certain property of the user's message, which would disclose some information about the message to the signer. They present a construction of a simulatable oblivious transfer protocol from so-called unique selective-failure blind signature schemes (in the random oracle model) for which the signature is uniquely determined by the message. Since the main result of the work [CNS07] is the construction of oblivious transfer protocols, the

¹Ultimately, since the malicious signer causes the abort, this can be seen as a more general case of signer aborts.

authors note that Chaum’s scheme [Cha83b] and Boldyreva’s protocol [Bol03a] are examples of such selective-failure blind schemes, but do not fully explore the relationship to (regular) blindness.

Hazay et al. [HKKL07a] present a concurrently-secure blind signature scheme and, as part of this, they also introduce a notion called a-posteriori blindness. This notion considers blindness of multiple executions between the signer and the user (as opposed to two sessions as in the basic case), and addresses the question how to deal with executions in which the user cannot derive a signature. However, the definition of a-posteriori blindness is neither known to be implied by ordinary blindness, nor implies it ordinary blindness (as sketched in [HKKL07a]). Thus, selective-failure blindness does not follow from this notion.

Aborts of players have also been studied under the notion of fairness in two-party and multi-party computations, especially for the exchange of signatures, e.g., [Gol04, ASW98, GMPY06]. Fairness should guarantee that one party obtains the output of the joint computation if and only if the other party receives it. Note, however, that in case of blind signatures the protocol only provides a one-sided output to the user (namely, the signature). In addition, solutions providing fairness usually require extra assumptions like a trusted third party in case of disputes, or they add a significant overhead to the underlying protocol.

Our Results in this Section. We pick up the idea of selective-failure blindness to deal with signer aborts and expand the work of Camenisch et al. [CNS07] towards its relationship to blindness and further constructions of such schemes. We first show that selective-failure blindness is indeed a strictly stronger notion than regular blindness. In [FS09] we also extend the notion of selective-failure blindness to multiple executions, particularly addressing aborts of a subset of executions. Moreover, we give two possible definitions for the multi-execution case and prove them to be equivalent. We then show that blindness in the basic case of two executions suffices to guarantee security in the case of many sessions and discuss the relation to a-posteriori blindness [HKKL07a]. We omit the definitions from this thesis because they are not required for our final transformation.

In Section 4.3 we present a general transformation which turns every secure blind signature scheme into a selective-failure blind scheme. Our transformation only requires an additional commitment of the message, which the user computes before the actual protocol starts and which the user then uses in the original protocol instead of the message itself.² Since the commitment is non-interactive, our transformation inher-

²This idea has been conjectured by Hazay et al. [HKKL07a] to also work for a-posteriori blindness. We are not aware of any formal claim or proof in the literature that uses a commitment indeed, provides security against aborts.

its important characteristics of the underlying protocol like the number of moves and concurrent security.

It should be noted, though, that the transformation destroys uniqueness (i.e., that each message has only one valid signature per key pair), as required by [CNS07] to derive oblivious transfer from such blind signatures. However, we show that our transformation is still applicable if we modify the oblivious transfer protocol of [CNS07] slightly. Hence, we can now easily obtain an adaptive oblivious transfer from any unique blind signature scheme such that the protocol is simulatable in presence of failures. Put differently, we show that selective-failure blindness is not necessary to obtain such oblivious transfer protocols, but uniqueness is sufficient. We note that like the original protocol in [CNS07] this result holds in the random oracle model.

4.2 Selective-Failure Blindness

In this section we review the definition of selective-failure blindness and show that selective-failure blindness is a strictly stronger requirement than the basic blindness property.

4.2.1 Definition

Camenisch et al. [CNS07] put forward the notion of *selective-failure blindness*, which roughly says that a malicious signer \mathcal{S}^* cannot force the user's algorithm \mathcal{U} to abort (based on the specific message) and that blindness should also hold in case the signer is able to learn that some executions have aborted. This is formalized by informing \mathcal{S}^* which instance has aborted (i.e., if the left, the right, or both user instances have failed):

Definition 4.2.1 (Selective-Failure Blindness). *A blind signature scheme $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ is called selective-failure blind if it is unforgeable (as in Definition 2.3.1) and if for any efficient algorithm \mathcal{S}^* (which works in modes *find*, *issue*, and *guess*) the probability that the experiment $\text{SFBlind}_{\mathcal{S}^*}^{\text{BS}}(n)$ evaluates to 1 is negligibly close to 1/2, where*

Experiment $\text{SFBlind}_{\mathcal{S}^*}^{\text{BS}}(n)$

$(pk, m_0, m_1, \beta_{\text{find}}) \leftarrow \mathcal{S}^*(\text{find}, 1^n)$

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

$\beta_{\text{issue}} \leftarrow \mathcal{S}^*(\langle \cdot, \mathcal{U}(pk, m_b) \rangle^1, \langle \cdot, \mathcal{U}(pk, m_{1-b}) \rangle^1)(\text{issue}, \beta_{\text{find}})$

and let σ_b, σ_{1-b} denote the (possibly undefined) local outputs

of $\mathcal{U}(pk, m_b)$ resp. $\mathcal{U}(pk, m_{1-b})$.
define answer as: left if only the first execution has failed,
right if only the second execution has failed,
both if both executions have failed,
and (σ_b, σ_{1-b}) otherwise.
 $b^* \leftarrow \mathcal{S}^*(\text{guess}, \text{answer}, \beta_{\text{issue}})$
Return 1 iff $b = b^*$.

4.2.2 Relation to Regular Blindness

We first prove formally the fact that selective-failure blindness implies regular blindness. Then we separate the notion by turning a secure blind signature scheme into a one which is still secure but provably not selective-failure blind.

Proposition 4.2.2. *Every selective-failure blind signature scheme BS_{SF} is also a secure blind signature scheme.*

Proof. Assume that there exists an adversarial controlled signer \mathcal{A} breaking blindness with noticeable probability. Then we construct an attacker \mathcal{S}^* breaking selective-failure blindness (with noticeable probability). The adversary \mathcal{S}^* invokes a black-box simulation of \mathcal{A} . Whenever \mathcal{A} interacts with the user algorithm (as described in the experiment), \mathcal{S}^* forwards the messages (in both directions). At the end of the protocol \mathcal{S}^* is informed if and which of the protocol executions have failed. In case that at least one of the user instances has aborted, the adversary \mathcal{S}^* forwards the pair (\perp, \perp) to \mathcal{A} , and otherwise, \mathcal{S}^* obtains two signatures and hands them to \mathcal{A} . In both cases, \mathcal{A} replies with a bit b^* , which \mathcal{S}^* too outputs and stops.

A straightforward analysis shows that the success probabilities of \mathcal{S}^* and \mathcal{A} in the corresponding experiment are identical. Moreover, the notions of unforgeability are the same in both definitions. \square

Proposition 4.2.3. *If there exists a secure blind signature scheme BS , then there exists a secure blind signature scheme $\text{BS}_{\overline{\text{SF}}}$ which is not selective-failure blind.*

Proof. We modify BS slightly into a scheme $\text{BS}_{\overline{\text{SF}}}$ which is identical to BS , except that we modify the key generation algorithm by adding a break condition into the user algorithm. More precisely, let $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ be a secure blind signature scheme. We define the new blind signature scheme $\text{BS}_{\overline{\text{SF}}}$:

KEYGEN. $\text{KG}_{\overline{\text{SF}}}$ first sets $m_{\text{max}} = 1^n$ as the maximum of the lexicographical order over n -bit strings. It then executes the key generation algorithm of the underlying blind signature scheme $(sk, pk) \leftarrow \text{KG}(1^n)$ and returns $(sk_{\overline{\text{SF}}}, pk_{\overline{\text{SF}}}) = (sk, (pk, m_{\text{max}}))$.

SIGNING PROTOCOL. The interactive signing protocol remains unchanged except for one modification. The user algorithm checks *after* the last move of the protocol (and after computing the signature σ) that $m \leq m_{\max}$ and, if so, it returns the signature σ , and \perp otherwise.

VERIFICATION. The verification algorithm returns the result of $\forall f$.

The modified scheme is clearly complete, as the case $m > m_{\max}$ for an honest signer never occurs and because the initial protocol is complete. Obviously, if the blind signature scheme BS is unforgeable, then $\text{BS}_{\overline{\text{SF}}}$ is also unforgeable. This is easy to see as the malicious user may simply ignore the break condition.

Concerning blindness, first note that the malicious signer \mathcal{S}^* is allowed to choose the public key and thus to pick some other value m_{\max}^* . As a malicious signer \mathcal{S}^* is not informed which of the executions has failed (if any), setting some other value m_{\max}^* than the predetermined maximum and possibly causing an abort does not lend any additional power to \mathcal{S}^* . To see this, note that the user algorithm does not abort prematurely if $m > m_{\max}$. Hence, from the (malicious) signer's point of view, the interaction is indistinguishable from an honest execution. It therefore follows that $\text{BS}_{\overline{\text{SF}}}$ still satisfies blindness.

We finally show that the modified scheme does not fulfill selective-failure blindness. Consider a malicious signer \mathcal{S}^* in experiment $\text{SFBlind}_{\mathcal{S}^*}^{\text{BS}}(n)$. In the first step the adversary \mathcal{S}^* computes a key pair $(sk, pk) \leftarrow \text{KG}(1^n)$, it sets $m_{\max}^* = 10^{n-1}$ and picks two messages $m_0 = 0^n, m_1 = 1^n$ such that $m_0 \leq m_{\max}^* < m_1$. It outputs a public key $pk_{\overline{\text{SF}}} = (pk, m_{\max}^*)$ together with the message m_0, m_1 as defined in the first step of the experiment. Next, \mathcal{S}^* has black-box access to two honest user instances (as described in experiment $\text{SFBlind}_{\mathcal{S}^*}^{\text{BS}}(n)$) where the first algorithm takes as input $(pk_{\overline{\text{SF}}}, m_b)$ and the second user algorithm receives $(pk_{\overline{\text{SF}}}, m_{1-b})$. In both executions \mathcal{S}^* acts like the honest signer with key $sk_{\overline{\text{SF}}} = sk$. Then \mathcal{S}^* is eventually informed which of the executions has failed, i.e., receives *left* or *right* (as \mathcal{S}^* has access to honest user instances, the case where both executions fail cannot occur by the completeness condition). The adversary \mathcal{S}^* returns $b^* = 1$ if the *left* instance has failed, otherwise it returns $b^* = 0$.

It follows straightforwardly that the adversary \mathcal{S}^* succeeds in predicting b with probability 1. □

4.3 From Blindness to Selective-Failure Blindness

In this section we show how to turn every secure blind signature scheme BS into a selective-failure blind signature scheme BS_{SF} . The high-level idea is to modify BS

slightly into BS_{SF} by executing BS with a non-interactive commitment Com of the message m (instead of the message itself).

Definition 4.3.1 (Commitment Scheme). *A (non-interactive) commitment scheme consists of a tuple of efficient algorithms $\mathcal{C} = (\text{KG}_{\text{com}}, \text{Com}, \text{Vf}_{\text{com}})$, where*

KEY GENERATION. *The algorithm $\text{KG}_{\text{com}}(1^n)$ takes as input the security parameter 1^n and outputs a key pk_{com} .*

COMMITMENT PHASE. *The input of the algorithm Com is a key pk_{com} and a message $m \in \{0, 1\}^n$; It outputs $(\text{Decom}, \text{Com}) \leftarrow \text{Com}(pk_{\text{com}}, m)$.*

VERIFICATION. *The algorithm $\text{Vf}_{\text{com}}(pk_{\text{com}}, m, \text{Decom}, \text{Com})$ outputs a bit b .*

It is assumed that the commitment scheme is complete, i.e., for any $n \in \mathbb{N}$, any $pk_{\text{com}} \leftarrow \text{KG}_{\text{com}}(1^n)$, for any message $m \in \{0, 1\}^n$, and any $(\text{Decom}, \text{Com}) \leftarrow \text{Com}(pk_{\text{com}}, m)$ we have $\text{Vf}_{\text{com}}(pk_{\text{com}}, m, \text{Decom}, \text{Com}) = 1$.

Security of commitment schemes is defined by *secrecy* and *unambiguity*. Secrecy guarantees that the receiver cannot learn the message from the commitment and unambiguity says that the sender cannot change the message anymore once the commitment phase is over. Here we use a slightly different way to define secrecy compared to the literature, but it is easy to see by a hybrid argument that our definition is equivalent:

Definition 4.3.2 (Secure Commitment). *A (non-interactive) commitment scheme $\mathcal{C} = (\text{KG}_{\text{com}}, \text{Com}, \text{Vf}_{\text{com}})$ is secure if the following holds:*

SECRECY. *For any efficient algorithm R_{com}^* (working in modes *find* and *guess*) the probability that experiment $\text{Secrecy}_{\text{R}_{\text{com}}^*}^{\mathcal{C}}(n)$ evaluates to 1 is negligibly close to $1/2$, where*

Experiment $\text{Secrecy}_{\text{R}_{\text{com}}^*}^{\mathcal{C}}(n)$:
 $(m_0, m_1, pk_{\text{com}}, \beta_{\text{find}}) \leftarrow \text{R}_{\text{com}}^*(\text{find}, 1^n)$
 $b \leftarrow \{0, 1\}$
 $\text{Com}_b \leftarrow \text{Com}(pk_{\text{com}}, m_b)$ and $\text{Com}_{1-b} \leftarrow \text{Com}(pk_{\text{com}}, m_{1-b})$
 $b^* \leftarrow \text{R}_{\text{com}}^*(\text{guess}, \beta_{\text{find}}, \text{Com}_0, \text{Com}_1)$
 Return 1 iff $b = b^*$.

UNAMBIGUITY. *For any efficient algorithm S_{com}^* the probability that experiment $\text{Unab}_{\text{S}_{\text{com}}^*}^{\mathcal{C}}(n)$ evaluates to 1 is negligible, where*

Experiment $\text{Unab}_{\text{S}_{\text{com}}^*}^{\mathcal{C}}(n)$:
 $pk_{\text{com}} \leftarrow \text{KG}_{\text{com}}(1^n)$
 $(m, m', \text{Decom}, \text{Decom}') \leftarrow \text{S}_{\text{com}}^*(pk_{\text{com}})$

Return 1 iff
 $\text{Vf}_{\text{Com}}(pk_{\text{com}}, m, \text{Decom}, \text{Com}) = 1$ and
 $\text{Vf}_{\text{Com}}(pk_{\text{com}}, m', \text{Decom}', \text{Com}) = 1$ as well as $m \neq m'$.

Note that such commitment schemes exist under standard assumptions like pseudorandom generators [Nao91] or hash functions [DPP97]. In order to use a commitment in a blind signature scheme —which we defined to take messages of n bits— we need that the commitment scheme is *length-invariant*, meaning that for n -bit messages the commitment itself is also n bits. This can always be achieved by using a collision-resistant hash function (with n bits output) on top.

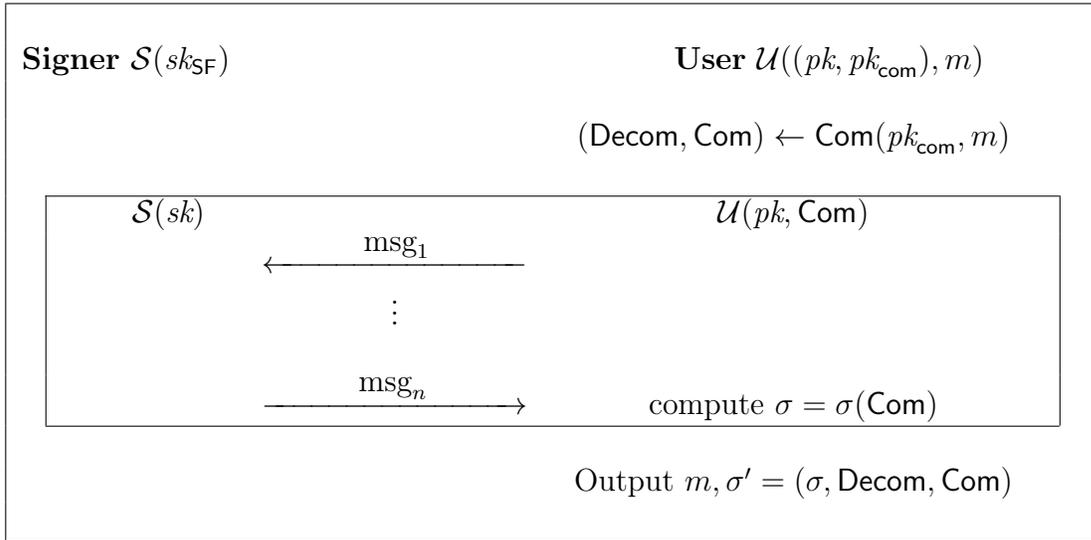


Figure 4.1: Issue protocol of the blind signature scheme BS_{SF} .

Construction 4.3.3 (Selective-Failure Blind Signature Scheme BS_{SF}). We denote by $\text{BS} = (\text{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \text{Vf})$ a blind signature scheme and let \mathcal{C} be a length-invariant commitment scheme. We define the blind signature scheme BS_{SF} through the following three procedures:

KEY GENERATION. The key generation algorithm $\text{KG}_{\text{SF}}(1^n)$ executes the key generation algorithm of the blind signature scheme BS , e.g., $(sk, pk) \leftarrow \text{KG}(1^n)$. It also runs the key generation algorithm of the commitment scheme, e.g., $pk_{\text{com}} \leftarrow \text{KG}_{\text{com}}(1^n)$. It returns the private-key $sk_{\text{SF}} = sk$ and the public-key $pk_{\text{SF}} = (pk, pk_{\text{com}})$.

SIGNATURE ISSUE PROTOCOL. The interactive signature issue protocol for message $m \in \{0, 1\}^n$ is described in Figure 4.1.

SIGNATURE VERIFICATION. *The verification algorithm $\text{Vf}_{\text{SF}}(pk_{\text{SF}}, m, \sigma')$ parses σ' as $(\sigma, \text{Decom}, \text{Com})$; it returns 1 iff*

$$\text{Vf}(pk, \sigma, \text{Com}) = 1 \quad \text{and} \quad \text{Vf}_{\text{Com}}(pk_{\text{com}}, m, \text{Decom}, \text{Com}) = 1.$$

Theorem 4.3.4. *If BS is a secure blind signature scheme and \mathcal{C} is a secure, length-invariant commitment scheme, then the scheme BS_{SF} in Construction 4.3.3 is a selective-failure blind signature scheme.*

We note that, if the starting blind signature scheme provides statistical blindness, and the commitment scheme is also statistically-hiding, then the derived protocol achieves selective-failure blindness in a statistical sense. This can be seen from the proof of Theorem 4.3.4, which is split into two claims, (1) covering unforgeability and (2) selective-failure blindness:

CLAIM 1: BS_{SF} is unforgeable.

Intuitively, in the proof we distinguish between two cases. The first case occurs if the adversary \mathcal{U}^* succeeds in outputting $k+1$ valid pairs of the form $m_i, \sigma'_i = (\sigma_i, \text{Decom}_i, \text{Com}_i)$ such that the commitments Com_i are pairwise different. However, we can then break the unforgeability of the underlying blind signature scheme BS . In the second case \mathcal{U}^* succeeds and at least two commitments $\text{Com}_i, \text{Com}_j$ (with $i \neq j$) are identical. But then we can break the unambiguity of the commitment scheme \mathcal{C} .

Proof. Assume to the contrary that the resulting selective-failure blind signature scheme BS_{SF} is *not* unforgeable. Then there exists an adversary \mathcal{U}^* breaking unforgeability with noticeable probability, i.e., on input pk_{SF} the algorithm \mathcal{U}^* returns $k+1$ valid signatures $\sigma'_i = (\sigma_i, \text{Decom}_i, \text{Com}_i)$ for messages m_i after at most k interactions with the honest signer \mathcal{S} . Note that here we do not deal with user aborts and count any initiated interaction; the case of counting only completed interactions is taken care of in the next section.

We first take a look at the success probability of \mathcal{U}^* , where we have

$$\psi(n) := \text{Prob} \left[\text{Forge}_{\mathcal{U}^*}^{\text{BS}_{\text{SF}}}(n) = 1 \right]$$

where $\psi(n)$ is noticeable. This probability can be separated according to the two exclusive events that \mathcal{U}^* succeeds and all commitments Com_i are different, with the corresponding probability denoted by $\psi_0(n)$, and into the case where \mathcal{A}_{SF} succeeds and at least two commitments are identical (with probability $\psi_1(n)$) According to our assumption that $\psi(n)$ is noticeable, $\psi_0(n)$ or $\psi_1(n)$ (or both) must be noticeable.

We next construct out of \mathcal{U}^* algorithms \mathcal{A}_{UNF} and \mathcal{A}_{UNA} against unforgeability of BS and unambiguity of the commitment scheme \mathcal{C} .

Attacking Unforgeability. The adversary \mathcal{A}_{UNF} takes as input the public key pk of the blind signature scheme BS and works as follows. It executes the key generation algorithm of the commitment scheme $pk_{\text{com}} \leftarrow \text{KG}_{\text{com}}(1^n)$ and runs a black-box simulation of \mathcal{U}^* on input $pk_{\text{SF}} = (pk, pk_{\text{com}})$. The signer instances in the attack of \mathcal{U}^* are simulated with the help of the external signer instances accessible by \mathcal{A}_{UNF} , i.e., adversary \mathcal{A}_{UNF} relays the communication between \mathcal{U}^* and its signer instance oracle $\mathcal{S}(sk)$ (as described in experiment $\text{Forge}_{\mathcal{U}^*}^{\text{BS}}$). When \mathcal{U}^* finishes its attack, it outputs $k + 1$ message-signatures pairs m_i, σ'_i after at most k interactions. Now \mathcal{A}_{UNF} parses each σ'_i as $(\sigma_i, \text{Decom}_i, \text{Com}_i)$, returns the $k + 1$ pairs Com_i, σ_i , and stops.

Assume that $\psi_0(n)$, the probability that \mathcal{U}^* succeeds and all Com_i 's are distinct, is noticeable. Then, since the simulation is perfect from the viewpoint of \mathcal{U}^* , adversary \mathcal{A}_{UNF} succeeds in outputting $k + 1$ valid pairs Com_i, σ_i for distinct “messages” Com_i with noticeable probability, too, contradicting the unforgeability property of the underlying blind signature scheme. Note also that the numbers of initiated and completed executions are identical in both cases.

Attacking Unambiguity. In order to break the unambiguity of \mathcal{C} , the adversary \mathcal{A}_{UNA} takes as input the public key pk_{com} of the commitment scheme \mathcal{C} and works as follows. It executes the key generation algorithm of the blind signature scheme $(sk, pk) \leftarrow \text{KG}(1^n)$ as well as the honest signer algorithms $\mathcal{S}(sk)$ and it runs a black-box simulation of \mathcal{U}^* on input $pk_{\text{SF}} = (pk, pk_{\text{com}})$. Note that running the program of the honest signer on input sk simulates each execution with a signer instance. Algorithm \mathcal{U}^* eventually returns $k + 1$ message-signature pairs (m_i, σ'_i) after at most k interactions with \mathcal{S} . The adversary \mathcal{A}_{UNA} then checks if there are valid signatures with $\text{Com}_i = \text{Com}_j$ for some $i \neq j$ and, if so, outputs two tuples $(m_i, \text{Decom}_i, \text{Com}_i), (m_j, \text{Decom}_j, \text{Com}_j)$ such that $m_i \neq m_j$ and $\text{Com}_i = \text{Com}_j$. If not, it outputs a failure message.

For the analysis note that the simulation again perfectly mimics the original attack of \mathcal{U}^* . Hence, if $\psi_1(n)$ is noticeable, then such $\text{Com}_i = \text{Com}_j$ with valid decommitments for $m_i \neq m_j$ appear with noticeable probability, and the commitment adversary \mathcal{A}_{UNA} therefore finds an ambiguous commitment with this probability, too. But this clearly violates the security of the commitment scheme \mathcal{C} . \square

CLAIM 2: BS_{SF} is selective-failure blind.

The high-level idea of the proof is as follows. We again distinguish between two cases. In the first case the adversary \mathcal{A}_{SF} succeeds with noticeable probability and both message-signature pairs are valid. But then we show how to break the blindness property of the underlying blind signature scheme BS . We next argue that in the case where \mathcal{A}_{SF} succeeds with noticeable probability and forces at least one of the user algorithms to fail, then we are able to break the secrecy of the commitment scheme (because then the only information available to the signer are the commitments of the messages).

Proof. Assume towards contradiction that the resulting blind signature scheme BS_{SF} is *not* selective-failure blind, and that there exists a successful adversary \mathcal{A}_{SF} against selective-failure blindness. Let

$$\delta(n) := \text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1] = \frac{1}{2} + \epsilon(n),$$

where $\epsilon(n) = \delta(n) - \frac{1}{2}$ is noticeable. We divide the success case according to the two exclusive events that \mathcal{A}_{SF} succeeds and that both message-signature pairs are valid (event **valid**) and into the case where \mathcal{A}_{SF} succeeds and at least one of the signatures is not valid (event $\neg\text{valid}$). Then,

$$\begin{aligned} \text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1] - \frac{1}{2} &= \text{Prob}[\text{valid}] \cdot (\text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \text{valid}] - \frac{1}{2}) \\ &\quad + \text{Prob}[\neg\text{valid}] \cdot (\text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \neg\text{valid}] - \frac{1}{2}). \end{aligned}$$

According to our assumption that $\delta(n)$ is noticeable, either the first term, denoted $\delta_0(n)$, or the second term $\delta_1(n)$ has to be noticeable (or both are noticeable). We next turn \mathcal{A}_{SF} into algorithms $\mathcal{A}_{\text{blind}}$ and \mathcal{A}_{com} against regular blindness and secrecy of the commitment scheme, respectively.

Attacking Blindness. The adversary $\mathcal{A}_{\text{blind}}$ works as follows. It runs a black-box simulation of \mathcal{A}_{SF} , which initially outputs two messages (m_0, m_1) together with a public key pk_{SF} . The adversary $\mathcal{A}_{\text{blind}}$ extracts pk and pk_{com} from pk_{SF} and calculates the commitments (and decommitments) $(\text{Decom}_0, \text{Com}_0) \leftarrow \text{Com}(pk_{\text{com}}, m_0)$ and $(\text{Decom}_1, \text{Com}_1) \leftarrow \text{Com}(pk_{\text{com}}, m_1)$. It outputs $\text{Com}_0, \text{Com}_1$ and pk . It is then given access to two user instances $\mathcal{U}(pk, \text{Com}_b)$ and $\mathcal{U}(pk, \text{Com}_{1-b})$ for a unknown bit b and relays the communication between these instances and \mathcal{A}_{SF} . If, at the end, at least one of the (external) user algorithms fails, then $\mathcal{A}_{\text{blind}}$ outputs a random bit and stops. Otherwise, it augments σ_0, σ_1 to $\sigma'_0 = (\sigma_0, \text{Decom}_0, \text{Com}_0)$ and $\sigma'_1 = (\sigma_1, \text{Decom}_1, \text{Com}_1)$

and returns the two signatures σ'_0, σ'_1 (obtained by the external user algorithms) to \mathcal{A}_{SF} . The final output of $\mathcal{A}_{\text{blind}}$ consists of the bit b^* returned by \mathcal{A}_{SF} .

Note that $\mathcal{A}_{\text{blind}}$ simulates the experiment $\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n)$ by executing the blindness experiment for the underlying blind signature scheme BS and by computing the commitments internally. Hence, the case where both message-signature pairs are valid is the one where experiment $\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n)$ is identical to experiment $\text{Blind}_{\mathcal{A}_{\text{blind}}}^{\text{BS}}(n)$. If one of the signatures is invalid, then $\mathcal{A}_{\text{blind}}$ returns a random bit. Therefore, the success probability of $\mathcal{A}_{\text{blind}}$ in experiment $\text{Blind}_{\mathcal{A}_{\text{blind}}}^{\text{BS}}(n)$ can be calculated as:

$$\begin{aligned}
& \text{Prob}[\text{Blind}_{\mathcal{A}_{\text{blind}}}^{\text{BS}}(n) = 1] \\
&= \text{Prob}[b = b^* \wedge \neg\text{valid}] + \text{Prob}[b = b^* \wedge \text{valid}] \\
&= \text{Prob}[b = b^* \mid \text{valid}] \cdot \text{Prob}[\text{valid}] + \text{Prob}[b = b^* \mid \neg\text{valid}] \cdot \text{Prob}[\neg\text{valid}] . \\
&= \text{Prob}[\text{valid}] \cdot \text{Prob}[b = b^* \mid \text{valid}] + \frac{1}{2} \cdot (1 - \text{Prob}[\text{valid}]) \\
&= \text{Prob}[\text{valid}] \cdot \text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \text{valid}] + \frac{1}{2} \cdot (1 - \text{Prob}[\text{valid}]) \\
&= \frac{1}{2} + \text{Prob}[\text{valid}] \cdot (\text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \text{valid}] - \frac{1}{2}) \\
&= \frac{1}{2} + \delta_0(n).
\end{aligned}$$

According to our assumption that $\delta_0(n)$ is noticeable it follows that $\mathcal{A}_{\text{blind}}$ breaks the blindness of the underlying blind signature scheme BS with noticeable probability. This, however, contradicts our assumption that BS is a *secure* blind signature scheme.

Attacking Secrecy of the Commitment. In order to break the secrecy of the commitment scheme \mathcal{C} , the adversary \mathcal{A}_{com} executes a black-box simulation of \mathcal{A}_{SF} , which initially outputs two messages (m_0, m_1) as well as a public key pk_{SF} . The adversary \mathcal{A}_{com} extracts the keys pk_{com} and pk from pk_{SF} and outputs $(m_0, m_1, pk_{\text{com}})$ for the secrecy experiment of the commitment scheme. It then receives two commitments $\text{Com}_0, \text{Com}_1$, one for message m_b and the other one for message m_{1-b} (without knowing which commitment corresponds to which message).

The adversary now runs (in the role of the honest user $\mathcal{U}(pk, \text{Com}_0)$ and $\mathcal{U}(pk, \text{Com}_1)$) the selective-failure blindness experiment with \mathcal{A}_{SF} . At the end of the issue protocol each user instance returns either a signature for the commitment or \perp . In the case that both user algorithms return a valid signature, then \mathcal{A}_{com} outputs a random bit b^* and stops. Otherwise, if both user algorithms have failed, then \mathcal{A}_{com} sends the value **both** to \mathcal{A}_{SF} . In the case that the first user algorithm has failed, then \mathcal{A}_{com} returns **left** to \mathcal{A}_{SF} and else (if the second user algorithm has failed), it forwards **right** to \mathcal{A}_{SF} . The final output of \mathcal{A}_{com} consists of the bit b^* returned by \mathcal{A}_{SF} .

The adversary \mathcal{A}_{com} simulates the experiment of selective-failure blindness perfectly, up to the point where it obtains the (possibly undefined) signatures. Given that at least one of them is invalid, the simulation corresponds to the case $\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n)$ (given $\neg\text{valid}$) for the same choice b as in the commitment experiment. Else, \mathcal{A}_{com} outputs a random bit. Thus,

$$\begin{aligned}
& \text{Prob} \left[\text{Secrecy}_{\mathcal{R}_{\text{Com}}}^{\mathcal{C}}(n) = 1 \right] \\
&= \text{Prob}[b = b^* \wedge \text{valid}] + \text{Prob}[b = b^* \wedge \neg\text{valid}] \\
&= \text{Prob}[b = b^* \mid \neg\text{valid}] \cdot \text{Prob}[\neg\text{valid}] + \text{Prob}[b = b^* \mid \text{valid}] \cdot \text{Prob}[\text{valid}] \\
&= \text{Prob}[\neg\text{valid}] \cdot \text{Prob}[b = b^* \mid \neg\text{valid}] + \frac{1}{2} \cdot (1 - \text{Prob}[\neg\text{valid}]) \\
&= \text{Prob}[\neg\text{valid}] \cdot \text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \neg\text{valid}] + \frac{1}{2} \cdot (1 - \text{Prob}[\neg\text{valid}]) \\
&= \frac{1}{2} + \text{Prob}[\neg\text{valid}] \cdot (\text{Prob}[\text{SFBlind}_{\mathcal{A}_{\text{SF}}}^{\text{BS}}(n) = 1 \mid \neg\text{valid}] - \frac{1}{2}) \\
&= \frac{1}{2} + \delta_1(n).
\end{aligned}$$

If $\delta_1(n)$ is noticeable, it follows that \mathcal{A}_{com} breaks the secrecy of the commitment scheme with noticeable probability, contradicting the security of \mathcal{C} . \square

4.4 Selective Failures and Adaptive Oblivious Transfer

Camenisch et al. [CNS07] also show how to construct an adaptive oblivious transfer protocol out of any unique selective-failure blind signature scheme (in the random oracle model). Roughly speaking, uniqueness means that each message has only one signature per public key. More formally, a blind signature scheme is *unique* [GO93, CNS07] if for every (possibly maliciously chosen) public key pk and every message $m \in \{0, 1\}^*$, there exists at most one signature $s \in \{0, 1\}^*$ such that $\forall f(pk, m, s) = 1$.

In this section we focus on the question whether our transformation turning every blind signature into one with selective-failure blindness is applicable. We have already mentioned in the introduction of this section that the initial commitment destroys uniqueness of the blind signature scheme because each message may have several valid signatures per key pair. Here we show that it is nonetheless possible to build an adaptive k -out-of- N oblivious transfer protocol out of *any* unique blind signature scheme by applying our transformation. The following construction is a modification of the protocol in [CNS07] and, because of the problems with uniqueness, we have to prove the security of this construction from scratch, digging also into the proof of selective-failure blindness for our transformation.

4.4.1 Simulatable Adaptive Oblivious Transfer

Oblivious transfer (OT), proposed by Rabin [Rab81], is an interactive protocol between a sender S and a receiver R . The sender in this protocol gets as input N messages m_1, \dots, m_N and the receiver R wishes to retrieve the message m_c . OT protocols must satisfy the following two security properties: firstly, the sender S does not find out the receiver's choice $c \in \{1, \dots, N\}$ and, secondly, the receiver only obtains m_c and does not gain any information about the other messages m_i for $i \neq c$. For adaptive k -out-of- N oblivious transfer, $OT_{k \times 1}^N$, the receiver requests k of these N messages in rounds where the i -th choice is based on the previously obtained messages. We refer the reader to [CNS07, NP05] for detailed informations. Following [CNS07] closely we define *adaptive oblivious transfer* more formally. An adaptive k -out-of- N oblivious transfer scheme $OT_{k \times 1}^N$ is a tuple of efficient algorithms (S_I, R_I, S_T, R_T) that consists of an initialization phase and a transfer phase. During the initialization phase the sender and the receiver perform an interactive protocol where the sender executes the algorithm S_I on input (m_1, m_2, \dots, m_N) and the receiver runs the algorithm R_I without any input. At the end of the initialization protocol both parties output some (local) state information, denoted by S_0 and R_0 , respectively.

Once the initialization phase is over, both parties engage in a transfer protocol. During the i -th transfer, where $1 \leq i \leq k$, the sender runs the algorithm $S_T(S_{i-1})$ to obtain some state information S_i whereas the receiver runs the $R_T(R_{i-1}, c_i)$ algorithm on input state information R_{i-1} and its choice c_i indicating which message it wishes to receive. The receiver obtains some state information R_i together with the retrieved message m'_{c_i} . A scheme is complete if $m'_{c_i} = m_{c_i}$ for all messages m_1, \dots, m_N , for all selections $c_1, \dots, c_k \in \{1, \dots, N\}$, and for all coin tosses of the algorithms. Roughly speaking, security of oblivious transfer demands that the receiver only learns the chosen messages (sender security) and the sender does not know which messages has been chosen (receiver security). In the following we briefly recall (partly verbatim) the security definitions by Camenisch et al. [CNS07]. In contrast to the definition of Naor and Pinkas [NP05] it employs the real-world/ideal-world paradigm for both sender and receiver security (simulatable oblivious transfer). This paradigm compares the execution of an OT protocol in the real-world with an *ideal implementation* (see for example [Can00]). In the real-world experiment, both parties jointly execute the interactive protocol, whereas in the ideal-world the functionality is realized through a trusted third party. Informally, security requires that the malicious receiver/sender gains in the real-world no more information than in the ideal-world. To capture failures one allows the ideal model sender to transmit a bit b , indicating whether the transfer should succeed or abort. We note that this bit is independent of the choice of the receiver, reflecting the fact that the abort should not depend on the receiver's input.

Real Experiment. We begin with the description of the real-world experiment that involves arbitrary sender and receiver algorithms S_{real}^* and R_{real}^* . The experiment

$$\text{Reals}_{S_{\text{real}}^*, R_{\text{real}}^*}(N, k, m_1, \dots, m_N, c_1, \dots, c_k)$$

works as follows. Algorithm S_{real}^* on input (m_1, \dots, m_N) interacts with R_{real}^* that obtains no input. Both parties generate an initial state S_0 and R_0 , respectively. Afterwards, the sender and the receiver perform k interactions. During the i -th execution, for $1 \leq i \leq k$, the sender and receiver interact by running $S_i \leftarrow S_{\text{real}}^*(S_{i-1})$ and $(R_i, m'_{c_i}) \leftarrow R_{\text{real}}^*(R_{i-1}, c_i)$ where $c_i \in \{1, \dots, N\}$ is a message index. It is understood that both algorithm update their state information to S_i and R_i , respectively. Observe that m'_{c_i} and m_{c_i} are not necessarily identical if either party cheats. The output of the experiment $\text{Reals}_{S_{\text{real}}^*, R_{\text{real}}^*}$ is the tuple (S_k, R_k) of the final state information.

Next, we define the behavior of the honest sender and honest receiver algorithm. That is, the honest sender algorithm S_{real} in an $\text{OT}_{k \times 1}^N$ scheme (S_I, S_T, R_I, R_T) takes as input a set of messages (m_1, \dots, m_N) , it runs the algorithm $S_I(m_1, \dots, m_N)$ in the initialization phase and runs the S_T algorithm in all following interactions. This algorithm always returns $S_k = \epsilon$ as its final state. The honest receiver algorithm R_{real} runs the algorithm R_I during the initialization phase, the algorithm R_T during the transfer phase and stops, outputting the list of received messages $R_k = (m'_{c_1}, \dots, m'_{c_k})$ as its final state.

Ideal Experiment. In the ideal-world experiment

$$\text{Ideals}_{S_{\text{ideal}}^*, R_{\text{ideal}}^*}(N, k, m_1, \dots, m_N, c_1, \dots, c_k),$$

the (possibly malicious) sender algorithm $S_{\text{ideal}}^*(m_1, \dots, m_N)$ generates N messages $(m'_1, m'_2, \dots, m'_N)$ and hands these over to the trusted party T . In each of the k transfers, T receives a bit b_i and afterwards an index c'_i of the (possibly cheating) receiver R_{ideal}^* . If $b_i = 1$ and $c'_i \in \{1, \dots, N\}$, then T sends the message $m'_{c'_i}$ to the receiver, and otherwise, \perp . The output of the experiment **Ideal** is the tuple (S_i, R_i) of the final state information of S_{ideal}^* and R_{ideal}^* , respectively.

As above, we now define the honest ideal sender as well as the honest ideal receiver. The honest ideal sender $S_{\text{ideal}}(m_1, \dots, m_N)$ sends the messages m_1, \dots, m_N to the trusted party T in the initialization phase, during the i -th transfer it hands the bit $b_i = 1$ over to T and outputs $S_k = \epsilon$ as its final state. The honest ideal-world receiver R_{ideal} submits its real choice (c_1, c_2, \dots, c_k) to the trusted party and outputs the obtained messages $R_k = (m'_1, m'_2, \dots, m'_k)$ as its final state.

SENDER'S SECURITY. We say that an $\text{OT}_{k \times 1}^N$ is *sender secure* if for any efficient cheating real-world receiver R_{real}^* there exists an efficient ideal-world receiver R_{ideal}^* such that for any polynomial $N_q(n)$, any $N \in \{1, \dots, N_q(n)\}$, any message

m_1, m_2, \dots, m_N , and for any choice $c_1, c_2, \dots, c_k \in \{1, \dots, N\}$ with $k \in \{1, \dots, N\}$, the advantage for any efficient distinguisher D in distinguishing the distributions

$$\text{Real}_{S_{\text{real}}, R_{\text{real}}^*}(N, k, m_1, \dots, m_N, c_1, \dots, c_k)$$

and

$$\text{Ideal}_{S'_{\text{ideal}}, R'^*_{\text{ideal}}}(N, k, m_1, \dots, m_N, c_1, \dots, c_k)$$

is negligible in n .

RECEIVER'S SECURITY. We say that an $\text{OT}_{k \times 1}^N$ is *receiver secure* if for any efficient real-world malicious sender S_{real}^* , there exists an efficient ideal-world sender S'_{ideal}^* such that for any polynomial $N_m(q)$, any $N \in \{1, \dots, N_m(q)\}$, any message m_1, m_2, \dots, m_N , for any choice $c_1, c_2, \dots, c_k \in \{1, \dots, N\}$ with $k \in \{1, \dots, N\}$, the advantage for any efficient distinguisher D in distinguishing the distributions

$$\text{Real}_{S_{\text{real}}^*, R_{\text{real}}}(N, k, m_1, \dots, m_N, c_1, \dots, c_k)$$

and

$$\text{Ideal}_{S'^*_{\text{ideal}}, R'_{\text{ideal}}}(N, k, m_1, \dots, m_N, c_1, \dots, c_k)$$

is negligible in n .

4.4.2 Construction

Our construction, depicted in Figure 4.2, is a modification of the $\text{OT}_{k \times 1}^N$ protocol of Camenisch et al. and consists of a black-box construction using *any* unique (not necessarily selective-failure) blind signature scheme. The sender in the first step of the protocol generates a key-pair for the blind signature scheme and sends it to the receiver. The receiver, in return, hands N distinct commitments (for values $1, 2, \dots, N$, represented as n -bit-strings each) over to the sender. These commitments serve as “messages” for the signature generation. Note that distinctiveness of the commitments holds with high probability by the binding property.

After the sender has verified that all commitments are distinct, it encrypts each message in its database by XOR-ing the message m_i with $H(i, s_i)$, where i is the index of the i -th commitment Com_i and s_i is the unique signature of message Com_i under pk . The sender can easily compute this signature locally by running the signature issue protocol with the help of the signing key and an honest user instance for “message” Com_i .

After having finished the initialization phase, both parties engage in a transfer phase that consists of a run of the unique blind signature scheme. In the case that the receiver wishes to obtain the i -th message m_i , then it has to choose the commitment Com_i (as the message to be signed) during the signature issue protocol.

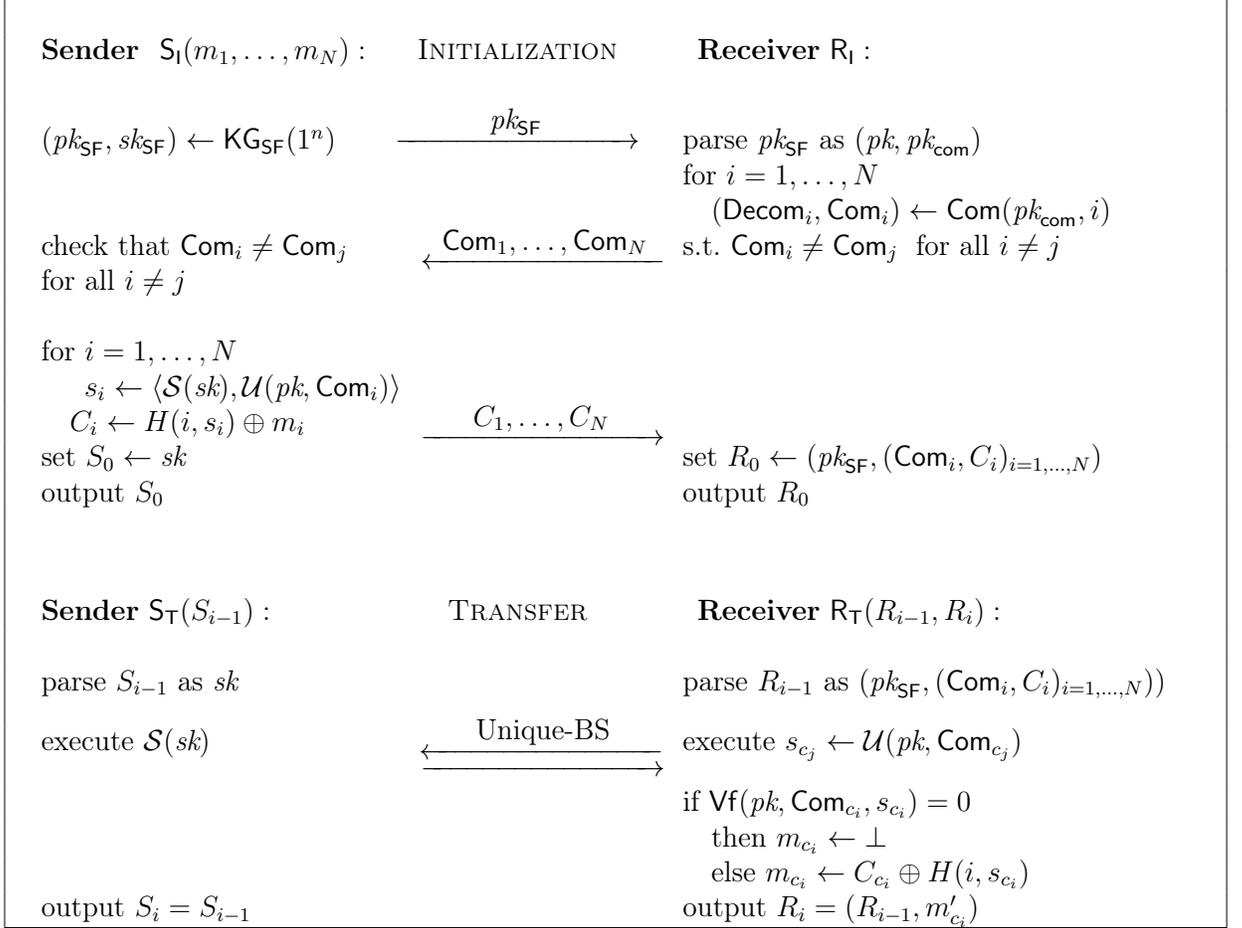


Figure 4.2: A k -out-of- N oblivious transfer protocol using a random oracle H and any unique blind signature scheme BS.

From a high-level point of view unforgeability guarantees that the receiver cannot receive more messages than interactions took place (sender’s security) and blindness guarantees that the sender cannot tell which message has been signed (receiver’s security).

4.4.3 Security of Our Construction

The security of the oblivious transfer protocol follows from the security of the unique blind signature scheme and from the security of the commitment scheme.

Theorem 4.4.1. *If the unique blind signature scheme BS is unforgeable then the $OT_{k \times 1}^N$ scheme depicted in Figure 4.2 is sender-secure in the random oracle model.*

Proof. In the following, we build for any malicious real-world receiver R_{real}^* an ideal-world receiver R_{ideal}^* that works as follows. The algorithm R_{ideal}^* first generates a key-pair $(sk_{\text{SF}}, pk_{\text{SF}}) \leftarrow \text{KG}_{\text{SF}}(1^n)$ for any unique blind signature scheme according to Construction 4.3.3. It forwards pk_{SF} to R_{real}^* and receives in return N commitments Com_i . Afterwards, R_{real}^* checks whether all commitments are distinct, and if so, it picks N random strings $C_i \leftarrow \{0, 1\}^n$ and sends these values to R_{real}^* to obtain the initial state R_0 .

During the transfer phase the ideal-world sender R_{ideal}^* simulates the honest signer of the unique blind signature scheme and engages in k executions of the signature issue protocol with R_{real}^* . In order to answer the random oracle queries, the algorithm R_{ideal}^* stores an initially empty associative array $\text{HT}[\cdot]$ together with a counter ctr . Whenever R_{real}^* invokes its random oracle $H(\cdot)$ on a value x , then the algorithm R_{ideal}^* returns $\text{HT}[x]$. In the case that this entry is undefined, then R_{ideal}^* proceeds as follows:

If $x = (i, s_i)$ and $\text{Vf}(pk, \text{Com}_i, s_i) = 1$ and $i \in [1, N]$ then
 $ctr \leftarrow ctr + 1$; If $ctr > k$, then abort
 Obtain m_i from the ideal functionality
 $\text{HT}[x] \leftarrow m_i \oplus C_i$
 else $\text{HT}[x] \leftarrow \{0, 1\}^\ell$.

Finally, at the end of the simulation, the algorithm R_{real}^* outputs its final state which R_{ideal}^* also outputs and stops.

It follows easily from the construction of R_{ideal}^* that the algorithm is efficient because R_{real}^* runs in polynomial time and because the overhead of the key generation, of the simulation of the honest signer algorithm, as well as the overhead of computing the verification equation, can all be performed efficiently. It is also clear that R_{ideal}^* performs a perfect simulation of the real-world experiment as long as R_{ideal}^* does not abort. Thus, there does *not* exist a distinguisher that is able to distinguish the real-world experiment from the ideal-world experiment with noticeable probability.

In the following we show that R_{ideal}^* does not cause R_{real}^* to abort with noticeable probability. To do so, let us assume towards contradiction that the algorithm R_{ideal}^* causes R_{real}^* to abort. But then we are able to build a forger F that breaks the unforgeability of BS with noticeable probability. Algorithm F performs a similar simulation of the environment but with two differences. Firstly, the algorithm F does not generate the keys for the blind signature scheme, but forwards the messages between its (external) signing oracle and R_{real}^* . Secondly, it does not abort if $\text{ctr} > k$, but it outputs all $k + 1$ tuples (Com_i, s_i) where (i, s_i) is the query sent by R_{real}^* to its random oracle. Observe that according to our protocol, all commitments have to be distinct and that R_{real}^* can engage in at most k transfer protocols. Thus, F outputs $k + 1$ different messages together with $k + 1$ valid signatures after at most k executions of the signature issue protocol. This, however, contradicts the assumption that the blind signature scheme BS is unforgeable. \square

Theorem 4.4.2. *If BS is a secure blind signature scheme and \mathcal{C} is a secure, length-invariant commitment scheme, then the $OT_{k \times 1}^N$ scheme depicted in Figure 4.2 is receiver-secure in the random oracle model.*

Proof. The proof follows the one in [CNS07] closely. We have to show that for any efficient cheating real-world sender S_{real}^* , there exists an efficient ideal-world sender S_{ideal}^* such that the outputs of both algorithms are (computational) indistinguishable. This ideal-world algorithm S_{ideal}^* works as follows. On input a set of messages (m_1, m_2, \dots, m_N) , algorithm S_{ideal}^* executes a black-box simulation of S_{real}^* on these messages and answers each random-oracle query by returning random values (but consistently). Let $pk_{\text{SF}} = (pk, pk_{\text{com}})$ be the first outgoing message produced by S_{real}^* . The attacker S_{ideal}^* now computes N distinct commitments $\text{Com}_i \leftarrow \text{Com}(pk_{\text{com}}, 0^n)$ and feeds them into S_{real}^* . Now, consider all random-oracle queries (i, s_i) with $1 \leq i \leq N$ made by S_{real}^* . For each query (i, s_i) , algorithm S_{real}^* checks whether $\text{Vf}(pk, \text{Com}_j, s_j) = 1$ for some $1 \leq j \leq N$ and if so, it stores $q_j \leftarrow H(j, s_j)$. During the last move of the initialization phase the algorithm S_{real}^* outputs the values C_1, \dots, C_N . Next, algorithm S_{ideal}^* sets $m'_i \leftarrow C_i \oplus q_i$ for all $1 \leq i \leq N$ if q_i is defined, or if q_i is not defined, it sets $m'_i \leftarrow \{0, 1\}^n$ to a random value and submits (m'_1, \dots, m'_N) to the trusted party.

In the following we have to handle the k transfer steps. To handle these queries, S_{ideal}^* sets $R_0 \leftarrow (pk_{\text{SF}}, \text{Com}_i, C_i)$ for $i = 1, \dots, N$ and simulates the environment of S_{real}^* during the i -th transfer by running $R_i \leftarrow R_{\text{T}}(R_{i-1}, 1)$, i.e., by always executing the honest receiver algorithm that wishes to receive the message m_1 . Observe that S_{ideal}^* does not get the choices (c_1, c_2, \dots, c_k) as input, thus it cannot run R_{T} on the real choice c_i . At the end of the i -th transfer phase, algorithm R_{T} may output \perp . In this case, algorithm S_{ideal}^* submits $b_i = 0$ to the trusted party indicating that this execution should be aborted, and otherwise, it sends $b_i = 1$.

Algorithm S_{ideal}^* simulates the transfer phase perfectly, i.e., queries of the form (i, s_i) with $1 \leq i \leq N$ and $\text{Vf}(pk, i, s) = 1$ are answered with $C_i \oplus m'_i$; other queries are answered with random values (but consistently). Finally, after having terminated k transfer queries, S_{real}^* outputs some state S_k which S_{ideal}^* also outputs and stops.

We use a standard hybrid argument (analogously to [CNS07]) to analyze the advantage of a distinguisher D in distinguishing between the experiments $\text{Real}_{S_{\text{real}}^*, R_{\text{real}}}$ and $\text{Ideal}_{S_{\text{ideal}}^*, R_{\text{ideal}}}$. By $S_{\text{ideal}, i}^*$ we denote an algorithm that simulates the environment of S_{real}^* identical to S_{ideal}^* except that it uses $R_i \leftarrow R_{\text{T}}(R_{i-1}, 1)$ for the first i transfer executions, and which uses $R_i \leftarrow R_{\text{T}}(R_{i-1}, c_i)$ for the remaining $k - i$ transfers. We further denote by **Game- i** the output of the corresponding experiment, i.e., it contains the final state of $S_{\text{ideal}, i}^*$ as well as the state of the honest ideal receiver $R_{\text{ideal}}(c_1, \dots, c_k)$ after interacting with the trusted party T . Obviously, **Game-0** corresponds to the real world experiment, i.e., **Game-0** = $\text{Real}_{S_{\text{real}}^*, R_{\text{real}}}$ and **Game- k** equals to our ideal-world experiment, i.e., **Game- k** = $\text{Ideal}_{S_{\text{ideal}}^*, R_{\text{ideal}}}$. The hybrid argument says that if there exists an efficient algorithm D that is able to distinguish the distributions $\text{Real}_{S_{\text{real}}^*, R_{\text{real}}}$ and $\text{Ideal}_{S_{\text{ideal}}^*, R_{\text{ideal}}}$ with noticeable advantage ϵ , then there must exist an index $0 \leq i \leq k$ such that D distinguishes **Game- i** and **Game- $(i + 1)$** with probability at least ϵ/k .

Next observe that the proof of Theorem 4.3.4, which says that every blind signature scheme is selective-failure blind when executed with an a-priori commitment, still holds. To see this note that we distinguish between two cases in this proof. Firstly, if no execution aborts then we can break blindness of the underlying blind signature scheme and, secondly, if at least one execution aborts then it is possible to break secrecy of the commitment scheme (see the proof of Claim 2). In the proof of Claim 2 the adversary receives the commitments of the messages (in random order) at the outset.

In the last step of the proof we have to show that we can use an algorithm D , that distinguishes between the games **Game- i** and **Game- $(i + 1)$** , to break selective-failure blindness of BS. To do so, we construct an algorithm \mathcal{A}_{SF} that runs a black-box simulation of S_{real}^* , that extracts the message and answers all random oracle queries as described for S_{ideal}^* . The algorithm \mathcal{A}_{SF} simulates the first j -th queries running algorithm $R_{\text{T}}(\cdot, 1)$, setting $m'_j = \perp$ if the transfer fails, and otherwise $m_j = m_{c_j}$.

During the $(i + 1)$ -th execution, the algorithm \mathcal{A}_{SF} behaves as follows. It outputs the tuple $(pk, m_0 = \text{Com}(pk_{\text{com}}, c_i), m_1 = \text{Com}(pk_{\text{com}}, 1))$ according to the first step of the (selective-failure) blindness experiment. In the next step of the experiment, \mathcal{A}_{SF} interacts with two honest user instances $\mathcal{U}(pk, m_b)$ and $\mathcal{U}(pk, m_{1-b})$ for a randomly chosen bit b in the following way. It relays the entire communication between S_{real}^* and the first user oracle $\mathcal{U}(pk, m_b)$, whereas it simply aborts the execution with the second oracle.

Algorithm \mathcal{A}_{SF} eventually receives the signatures (s_0, s_1) , encoding the answer **right** or **both**. In the case that the issuing in the first execution succeeds, i.e., if $s_0 \neq \perp$, it sets $m'_{i+1} = m_{c_{i+1}}$. Otherwise let $m'_{i+1} = \perp$. We remark that here selective-failure blindness (as opposed to regular blindness) is necessary in order to obtain the information about the left execution.

Analogously to our description above, \mathcal{A}_{SF} answers the remaining $i+2 \leq j \leq k$ transfers with the algorithm $\text{R}_\top(\cdot, c_j)$, setting $m'_j = m_{c_j}$ if the transfer succeeds, and otherwise, $m'_j = \perp$. Finally, when algorithm S_{real}^* outputs its final state S_k , then \mathcal{A}_{SF} runs the distinguisher D on input $(S_k, (m'_1, \dots, m'_k))$. Note that if $b = 0$, then this tuple is distributed according to **Game-i** and in the case that $b = 1$ it is distributed like **Game-(i + 1)**. Thus, algorithm \mathcal{A}_{SF} returns the output of D and wins the game with probability at least $1/2 + \epsilon/k$. \square

5 On the Impossibility of Blind Signatures From One-Way Permutations

5.1 Introduction

Several constructions of blind signature schemes are known based on specific number-theoretic assumptions in either the random oracle model [PS00b, Abe01b, BNPS03a, Bol03b, AO09] or the standard model [CKW04b, Oka06b, HK07, KZ08, AFG⁺10]. Blind signatures can also be build from oblivious transfer in a black-box way using techniques from secure two-party computation [LP07]. Constructions based on general assumptions are also known [JLO97b, Fis06b, HKKL07b, FS09], but the *minimal* assumptions needed to construct blind signatures are unclear. On the positive side, there exist constructions of blind signature schemes based on any trapdoor permutation [JLO97b, Fis06b]. Interestingly, the known constructions are all non black-box even in the honest-but-curious setting, with the construction by Juels, Luby, and Ostrovsky [JLO97b] applying techniques from secure two-party computation and the construction by Fischlin [Fis06b] using non-interactive zero-knowledge proofs. Since (standard) signature schemes can be constructed in a black-box way based on any one-way function [NY89, Rom90], there is no reason *a priori* to believe that blind signatures cannot be constructed from one-way functions also. Here, we settle the question in the negative:

Theorem 5.1.1 (Main Theorem). *There is no black-box construction of a blind signature scheme from one-way functions.*

Our result imposes no restrictions on the blind signature scheme, and applies even to schemes with imperfect completeness. Moreover, it is actually quite a bit more general than the above theorem indicates. In particular, our impossibility result applies also to constructions based on one-way permutations or random oracles, and even rules out constructions of blind signature schemes for 1-bit messages that achieve security only against honest-but-curious parties.

The proof of our impossibility result requires a careful combination of prior techniques in the area of black-box separations. At a high level, our basic framework is similar to the one used by Barak and Mahmoody-Ghidary in the context of black-box constructions of (standard) signature schemes from one-way functions [BMG07]. Our setting introduces several additional difficulties, however, not least of which is that we must also deal with the case of *interactive* protocols. Also, Barak and Mahmoody-Ghidary prove limits on the efficiency of the construction, whereas we are interested in proving impossibility. To deal with these complications, we also rely on techniques used in analyzing constructions of key-agreement protocols from one-way functions [IR89, BMG09].

5.2 Overview of Our Techniques

We begin by giving a high-level overview of our proof techniques. The full details are given in Sections 5.4.3 and 5.4.4. We consider an interactive signature issue protocol between a signer and a user. The input of the signer is a private key sk and the user's input is a public key pk and a message m . At the end of this protocol the user outputs a signature σ on the message m . Both of the algorithms are given black-box access to either a one-way function (OWF) or a one-way permutation (OWP). We assume that both players follow the protocol and are just honest-but-curious (semi-honest). We allow the players to be computationally unbounded, but require that they only query the one-way function (resp. permutation) a polynomial number of times.

In the setting of blind signatures, security demands that:

Unforgeability The user should not be able to output two valid signatures after interacting with the signer once. (More generally, the user should be unable to output $k + 1$ valid signatures on distinct messages after interacting with the signer k times.)

Blindness If the user executes the signature-issue protocol twice, once using a message m_0 and once using a message m_1 , then the signer should be unable to tell which order these executions were run. This should hold even if the signer is given both of the resulting signatures.

If we consider schemes that only achieve property (a), then it is well-known that these can be built in a black-box way from any OWF [Rom90] as this is just a standard signature scheme. On the other hand, schemes that are only blind but trivially forgeable can be constructed without any assumption letting the verification algorithm always output 1.

We show that if we wish to satisfy both conditions above then OWFs are not sufficient. To illustrate the main idea why this is true, consider the setting where both the user

and signer are given access to a random oracle. Let Q denote the oracle queries made by the signer in generating its public and private keys. Now consider two protocol executions in which the user first obtains a signature on the message m_0 and then obtains a signature on the message m_1 . Correctness intuitively requires that in each interaction the user learns sufficiently many of the queries in Q in order to be able to derive a valid signature. Unforgeability requires that the user does not learn “too many” of the queries in Q in each interaction; in particular, the user should not learn enough queries in the first interaction to derive a valid signature on m_1 . Finally, blindness implies that, from the point of view of the signer, the queries the user learns in the first interaction should be distributed identically to the queries the user learns in the second interaction. We show that all these requirements are in conflict.

More formally, we rely on results of [IR89, BMG09] showing that for any two-party protocol there is an algorithm Find_δ that takes as input a transcript of an execution of the protocol and outputs, with high probability, a set that contains every oracle query that was asked by both parties (“intersection queries”). Noting that the signer can run this algorithm, the blindness requirement thus implies that the set obtained by running Find_δ on the signature-issue protocol for m_0 must contain a set of intersection queries that are sufficient to derive a signature on the message m_1 . (Else the signer knows that the first execution could not possibly have been for m_1 .) We use this to construct a forger, which is a more efficient version of the one given in [BMG07]. Our forger runs a single protocol execution honestly to obtain a signature on m_0 , and then runs Find_δ to learn all the intersection queries. By what we have just said, this set will contain enough information to allow the forger to also compute a valid signature on m_1 . More precisely, the forger executes the following attack:

1. **INPUT KEY GENERATION.** The forger gets as input a public key pk .
2. **REQUEST SIGNATURE.** Engage in an interactive signature issue protocol using the honest user algorithm on the message 0 to get a signature σ_0 .
3. **LEARNING ORACLE QUERIES.** Let trans_0 be the transcript that corresponds to the above protocol execution. Then first run the algorithm Find_δ to compute the set \mathcal{I}_0 that contains the set of intersection queries and then also run the verification algorithm on the signature σ_0 corresponding to 0.
4. **SAMPLING A POSSIBLE TRANSCRIPT.** Conditioned on the knowledge learned from the previous step, i.e., all query/answer pairs that the user and the verification algorithm made and also all query/answer pairs determined by Find_δ , guess a secret key \tilde{sk} and an oracle $\tilde{\mathcal{O}}$ that agree with the information collected about the real secret key and oracle \mathcal{O} .
5. **FORGING.** Forge a signature for 1 using the key \tilde{sk} and the oracle $\tilde{\mathcal{O}}$, by running the signature issue protocol locally.

Using the blindness property of the scheme, we will show that the adversary outputs an additional message/signature pair in the last step with high probability showing that both blindness and unforgeability cannot hold simultaneously for such constructions.

5.2.1 Similarities and Differences to the Work of Barak and Mahmoody-Ghidary

From a technical point of view, our proof can be seen as the combination (plus some extensions) of the two techniques of Barak and Mahmoody-Ghidary [BMG07, BMG09]. Our technique differs from [BMG07] in the following aspects:

TWO-PARTY PROTOCOL. The obvious difference is that we consider an *interactive* signing protocol. In particular, this means that we have to consider two parties querying the OWP rather than a single one. Barak and Mahmoody-Ghidary consider deterministic signature scheme. Note that this assumption is in the case of regular signature scheme not a restriction. Here, however, we cannot assume that both parties are deterministic as blindness would no longer hold. As it turns out, having a second party that queries the OWP simplifies the proof as in the forgery game we basically learn only correct query/answer pairs from the user's side.

USEFULNESS PROPERTY. While we use the same “usefulness” property as in [BMG07], our proof that usefulness holds is very different from the analogous proof in their work: they assume a large message and argue that usefulness occurs for some pair of messages with high probability, whereas in our case we rely on blindness and show (roughly) that usefulness holds for any two messages with all but negligible probability. This allows us to simplify the attack and obtain a forger that makes only polynomially many oracle queries regardless of how many oracle queries the construction uses. (In the work of Barak and Mahmoody-Ghidary the number of queries made by the forger depends exponentially on the number of queries made by the construction.)

5.3 Blind Signatures Relative to an Oracle

5.3.1 Definition of Blind Signatures

We define blind signatures with black-box security with respect to an oracle \mathcal{O} . By $A^{\mathcal{O}}(x)$ we mean that an algorithm A on input x gets black-box access to \mathcal{O} . To simplify notation we often omit the oracle \mathcal{O} , but it is understood that all algorithms have access

to \mathcal{O} . In the following, we consider blind signatures for 1-bit messages; since we are proving impossibility, this only makes our results stronger.

Definition 5.3.1 (Oracle Blind Signature Scheme). *An oracle blind signature scheme is a tuple of polynomial-time algorithms $\text{BS} = (\text{KG}^{(\cdot)}, \mathcal{S}^{(\cdot)}, \mathcal{U}^{(\cdot)}, \text{Vf}^{(\cdot)})$, where for any oracle \mathcal{O} :*

KEY GENERATION. *The algorithm $\text{KG}^{\mathcal{O}}(1^n)$ for security parameter n generates a key pair (sk, pk) .*

SIGNATURE ISSUING. *The joint execution of algorithm $\mathcal{S}^{\mathcal{O}}(sk)$ and algorithm $\mathcal{U}^{\mathcal{O}}(pk, m)$ for message $m \in \{0, 1\}$ generates an output σ for the user and no output for the signer, we write this as $(\perp, \sigma) \leftarrow \langle \mathcal{S}^{\mathcal{O}}(sk), \mathcal{U}^{\mathcal{O}}(pk, m) \rangle$.*

VERIFICATION. *The algorithm $\text{Vf}^{\mathcal{O}}(pk, m, \sigma)$ outputs a bit b .*

We assume that the scheme has perfect completeness, i.e., for any $n \in \mathbb{N}$, any $(sk, pk) \leftarrow \text{KG}^{\mathcal{O}}(1^n)$, any message $m \in \{0, 1\}$, and any signature σ output by $\mathcal{U}^{\mathcal{O}}$ in the joint execution of $\mathcal{S}^{\mathcal{O}}(sk)$ and $\mathcal{U}^{\mathcal{O}}(pk, m)$, we have $\text{Vf}^{\mathcal{O}}(pk, m, \sigma) = 1$.

5.3.2 Security of Blind Signatures with Respect to an Oracle

Recall that security of blind signature schemes consists of unforgeability and blindness [JLO97b, PS96]. In this section, our definitions of security are weaker than those usually considered; since we show impossibility, this only strengthens our results.

In the definitions that follow we consider an execution of an oracle blind signature scheme BS relative to a random oracle \mathcal{O} . Since a random oracle is one-way with overwhelming probability, any construction of blind signatures from one-way functions must give an oracle blind signature scheme satisfying these definitions. We remark that our definitions consider unbounded adversaries who make polynomially many queries to \mathcal{O} ; however, we could have stated our definitions in terms of polynomial-time adversaries given access to an **NP** oracle.

Definition 5.3.2 (Secure Oracle Blind Signature Scheme). *An oracle blind signature scheme $\text{BS} = (\text{KG}^{\mathcal{O}}, \langle \mathcal{S}^{\mathcal{O}}, \mathcal{U}^{\mathcal{O}} \rangle, \text{Vf}^{\mathcal{O}})$ is secure if the following two properties hold:*

UNFORGEABILITY. *For any semi-honest algorithm $\mathcal{U}^{*\mathcal{O}}$ that invokes a single signature-issue protocol on the message 1 and that makes at most $\text{poly}(n)$ queries to the oracle \mathcal{O} , the probability that experiment $\text{Forge}_{\mathcal{U}^{*\mathcal{O}}}^{\text{BS}}(n)$ evaluates to 1 is negligible (in n), where*

Experiment $\text{Forge}_{\mathcal{U}^*}^{\text{BS}}(n)$:

Oracle $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is chosen at random

$(sk, pk) \leftarrow \text{KG}^{\mathcal{O}}(1^n)$

$(\sigma_0, \sigma_1) \leftarrow \mathcal{U}^{*(\mathcal{S}(sk), \cdot), \mathcal{O}}(pk)$ (where \mathcal{U}^* runs an honest execution of $\mathcal{U}(pk, 0)$ with \mathcal{S} and then outputs signatures of its choice)

Return 1 iff $\text{Vf}^{\mathcal{O}}(pk, 0, \sigma_0) = 1$ and $\text{Vf}^{\mathcal{O}}(pk, 1, \sigma_1) = 1$.

BLINDNESS. For any semi-honest algorithm $\mathcal{S}^{*\mathcal{O}}$ (working in modes *issue* and *guess*) that makes at most $\text{poly}(n)$ queries to \mathcal{O} , the probability that the following experiment $\text{Unblind}_{\mathcal{S}^{*\mathcal{O}}}^{\text{BS}}(n)$ evaluates to 1 is negligibly close to $1/2$, where

Experiment $\text{Unblind}_{\mathcal{S}^{*\mathcal{O}}}^{\text{BS}}(n)$

Oracle $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is chosen at random

$r \leftarrow \{0, 1\}^n$

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

$(sk, pk) \leftarrow \text{KG}^{\mathcal{O}}(1^n; r)$

$\text{st}_{\text{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}^{\mathcal{O}}(pk, b) \rangle^1, \langle \cdot, \mathcal{U}^{\mathcal{O}}(pk, 1-b) \rangle^1, \mathcal{O}}(\text{issue}, sk, pk, r)$ (where \mathcal{S}^* runs an honest execution of the protocol with each instance of \mathcal{U})
and let σ_b, σ_{1-b} denote the local outputs of $\mathcal{U}^{\mathcal{O}}(pk, m_b)$ resp. $\mathcal{U}^{\mathcal{O}}(pk, m_{1-b})$.

set $(\sigma_0, \sigma_1) = (\perp, \perp)$ if $\sigma_0 = \perp$ or $\sigma_1 = \perp$

$b^* \leftarrow \mathcal{S}^{*\mathcal{O}}(\text{guess}, \sigma_0, \sigma_1, \text{st}_{\text{issue}})$

return 1 iff $b = b^*$.

We say an oracle blind signature scheme $\text{BS}^{(\cdot)}$ is a *black-box construction from one-way functions* if for every \mathcal{O} that is one-way it holds that $\text{BS}^{\mathcal{O}}$ is secure.

5.3.3 Simplifying assumptions

To prove our result we make the following simplifying assumptions.

DETERMINISTIC SIGNER. We assume that the signing algorithm is deterministic. Note that this is not a restriction as a blind signature scheme with randomized signer \mathcal{S} can always be converted to a scheme with deterministic signer \mathcal{S}' by (1) including a key for a pairwise-independent hash function as part of the signer's private key (see page 29) ; (2) having the user send a random nonce as its first message in the signing protocol; and then (3) having \mathcal{S}' apply the hash function to the user's first message to generate random coins that it then uses to run \mathcal{S} .

PERFECT COMPLETENESS. In Section 5.4 we will only consider signature schemes achieving perfect completeness. We will show how to relax this requirement and rule out schemes with imperfect completeness in Section 5.5.1.

5.4 Attacking Black-Box Constructions of Blind Signatures

In this section we show that there is no black-box construction of blind signatures from one-way functions. To this end, we describe a cheating user \mathcal{U}^* who wins in the unforgeability game when the security of the blind signature scheme depends on a random function oracle $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}^n$. This rules out such constructions since such an oracle is one-way with overwhelming probability.

In the following we assume that the protocol proceeds in some fixed number of rounds. We also assume w.l.o.g. that no party queries \mathcal{O} twice on the same input. We say that any message sent from one party to the other party is a move. W.l.o.g. we assume that the signer sends all even moves and the user all odd ones.

5.4.1 Preliminaries — Notation and Oracles

We assume that \mathcal{O} is a *random oracle* $\mathcal{O} : \{0, 1\}^n \rightarrow \{0, 1\}^n$ that on input x outputs the value $\mathcal{O}(x)$. Let BS be an oracle blind signature scheme. For messages $(0, 1)$, let trans_0 (resp. trans_1) be a transcript of the execution with $\mathcal{U}(pk, 0)$ (resp. with $\mathcal{U}(pk, 1)$), and let σ_0 (resp. σ_1) be a corresponding signatures. Let $Q(\text{Vf}(0))$ (resp. $Q(\text{Vf}(1))$) denote the set of \mathcal{O} queries made by the verification algorithm $\text{Vf}^{\mathcal{O}}(pk, 0, \sigma_0)$ (resp. $\text{Vf}^{\mathcal{O}}(pk, 1, \sigma_1)$). Finally, let $Q(\mathcal{S}_0)$ (resp. $Q(\mathcal{S}_1)$) be the set of queries asked by the signer when interacting with $\mathcal{U}(pk, 0)$ (resp. $\mathcal{U}(pk, 1)$).

5.4.2 The Algorithm Find $_{\delta}$

Before proposing our attacker, we review a necessary lemma from Barak and Mahmoody-Ghidary [BMG09]. Informally, it states that for any two-party protocol where each party has access to a random oracle there exists an algorithm that, upon observing the transcript of the interaction, finds with high probability all the intersection queries (queries that have been asked by both parties). This result was first discovered by Impagliazzo and Rudich [IR89], and a more efficient protocol was given by Barak and Mahmoody-Ghidary [BMG09]. Formally, this result is given in the following lemma.

Lemma 5.4.1 ([BMG09]). *Let Π be a two-party (randomized) protocol where each party (Alice or Bob) asks at most q oracle queries to a random oracle. Then for every $0 < \delta < 1$, there is an algorithm Find_δ that has access to the messages sent between Alice and Bob and asks at most $(\frac{10^4 q^2}{\delta^2})$ oracle queries such that the queries made by Find_δ contain all the intersection queries of Alice and Bob with probability at least $1 - \delta$.*

We apply this lemma to the scenario of blind signatures defining the protocol Π as follows: Corresponding to any oracle blind signature scheme $\text{BS}^{(\cdot)}$, define the following two-party protocol Π between a signer \mathcal{S} and a user \mathcal{U} :

1. \mathcal{S} runs $(sk, pk) \leftarrow \text{KG}^{\mathcal{O}}(1^n)$ and sends pk to \mathcal{U} .
2. \mathcal{U} and \mathcal{S} then run the signature-issuing protocol on the message 1, at the end of which \mathcal{U} obtains a signature σ_0 .
3. \mathcal{U} runs $\text{Vf}^{\mathcal{O}}(pk, 0, \sigma_0)$.

For the remainder of Section 5.4, fix some δ and define Find_δ (as per Lemma 5.4.1) relative to the above protocol Π . Say the above protocol is run in the presence of a random oracle \mathcal{O} . If we let $Q(\mathcal{S}_\Pi)$ and $Q(\mathcal{U}_\Pi)$ denote the \mathcal{O} -queries made by each party during an execution of the above protocol that resulted in transcript trans , then Lemma 5.4.1 guarantees that, with high probability, that the set \mathcal{I} contains all the intersection queries, i.e.,

$$Q(\mathcal{S}) \cap Q(\mathcal{U}) \subseteq \mathcal{I},$$

Since the protocol Π is fixed, we omit this additional input in the following, i.e., $\text{Find}_\delta(\text{trans}) := \text{Find}_\delta(\Pi, \text{trans})$. Note that the message in Π is *fixed*, but the transcript might correspond to a different message. Due to the blindness, however, the success probability of the algorithm Find_δ is independent of the transcript.

5.4.3 From Blindness to Usefulness

In this section we study the question of what blindness means with respect to the set of intersection queries. The main observation is that due to blindness the set \mathcal{I} that contains all intersection queries must be somehow “independent” of the message. Recall that in the blindness game the semi-honest signer first outputs a public key together with two messages. Then, it interacts with two honest user instances in a random order. The task for the attacker is to predict which user had which message as input. Recall that the algorithm Find_δ (see Section 5.4.2) gets as input a transcript (i.e., all the messages exchanged between both parties) of a protocol execution and outputs a set that contains all intersection queries.

Now, consider two protocol executions and suppose that the set of intersection queries depends on the message. Then just by looking at these queries it is possible to determine the order of the messages. To formalize this intuition, consider a (semi-honest) signer \mathcal{S}^* in the blindness game. Since the attacker is semi-honest and by perfect completeness, the user instances get a valid signature. Then, the adversary obtains both signatures in the original order together with the transcripts of both executions. We now show that the set of queries that the verification algorithm makes to verify the second message 1 are already contained in the intersection queries of the first execution. We write \mathcal{I}_m , to indicate the set output when Find_δ is run on the protocol execution $\langle \mathcal{S}(sk), \mathcal{U}(pk, m) \rangle$. In the blindness game, where the message being signed is unknown, we use trans^0 to indicate the transcript of the first protocol execution $\langle \mathcal{S}(sk), \mathcal{U}(pk, m_b) \rangle$. trans^1 is defined similarly for the second execution. We let $\mathcal{I}^0 \leftarrow \text{Find}_\delta(\text{trans}^0)$ and define \mathcal{I}^1 similarly. We also define the sets $Q(\mathcal{S}^0)$ and $Q(\mathcal{S}^1)$ similarly for the queries made by the signer.

We now show that due to blindness all the intersection queries that occur when signing the message 1 are contained in the set \mathcal{I}_0 . That is, in the language of [BMG07], we show that 0 is “useful” for 1.

Lemma 5.4.2. *Let BS be an oracle blind signature scheme satisfying blindness. Consider an execution of the blindness experiment (cf. Definition 5.3.2), and let $Q(\text{KG})$, $Q(\mathcal{S}_b)$, trans_b , and $Q(\text{Vf}_b)$ be as defined above. Then with probability at least $1 - \delta - \text{negl}(n)$ over the random coins of the experiment it holds that*

$$Q(\text{Vf}_1) \cap (Q(\text{KG}) \cup Q(\mathcal{S}_0)) \subseteq \text{Find}_\delta(\text{trans}_0).$$

Proof. We first observe that with probability at least $1 - \delta$ we have

$$Q(\text{Vf}_1) \cap (Q(\text{KG}) \cup Q(\mathcal{S}_1)) \subseteq \text{Find}_\delta(\text{trans}_1).$$

This follows immediately from Lemma 5.4.1 and our definition of protocol Π in the previous section.

Consider now the following adversary \mathcal{S}^* :

1. \mathcal{S}^* runs the honest key-generation algorithm to obtain (sk, pk) . It records the \mathcal{O} -queries $Q(\text{KG})$ made during this step.
2. \mathcal{S}^* then runs the honest signing protocol with the first user instance. Let trans denote the transcript of this execution, and let $Q(\mathcal{S})$ denote the \mathcal{O} -queries made during this step.
3. \mathcal{S}^* then runs the honest signing protocol with the second user instance.

4. \mathcal{S}^* is given signatures σ_0, σ_1 on the messages 0 and 1, respectively. (By perfect completeness, both user instances always obtain valid signatures.) \mathcal{S}^* verifies σ_1 and records the \mathcal{O} -queries $Q(\mathbf{Vf}_1)$ made in doing so.
5. Finally, \mathcal{S}^* outputs 1 iff $Q(\mathbf{Vf}_1) \cap (Q(\mathbf{KG}) \cup Q(\mathcal{S})) \subseteq \text{Find}_\delta(\text{trans})$.

If $b = 1$, and so the first user instance represents an interaction with $\mathcal{U}(pk, 1)$, then $\text{trans} = \text{trans}_1$ and $Q(\mathcal{S}) = Q(\mathcal{S}_1)$ and so \mathcal{S}^* outputs 1 with probability at least $1 - \delta$. The blindness property thus implies that \mathcal{S}^* outputs 1 with probability at least $1 - \delta - \text{negl}(n)$ when $b = 0$ (and the first user instance represents an interaction with $\mathcal{U}(pk, 0)$). This concludes the proof. \square

5.4.4 Forging a Signature

Before presenting our forger, we begin by discussing the ideas behind our attack. The main observation is that due to the blindness of the signature scheme the intersection queries between the signer and user are somehow “independent” of the message. This was formalized in Lemma 5.4.2, where we showed that (with high probability)

$$Q(\mathbf{Vf}_1) \cap (Q(\mathbf{KG}) \cup Q(\mathcal{S}_0)) \subseteq \text{Find}_\delta(\text{trans}_0).$$

Intuitively, this means that all the “important” queries needed to verify a signature on the message ‘1’ must already be contained in the set of queries that are found when signing and verifying the message ‘0’. Thus, in the language of Barak and Mahmoody-Ghidary [BMG07], we have shown that 0 is “useful” for 1 with high probability. As in that paper, we use that property to show an attack.

The above condition seems to suggest that the set of intersection queries for ‘0’ is sufficient to generate a signature on ‘1’. However, this is not quite true. The problem is that there may be queries that the user makes with high probability when generating and verifying a signature for 1 that are not in the set $\text{Find}_\delta(\text{trans}_0)$; this could cause technical problems because our forger must get the answers to these queries right when constructing a forged signature. For a concrete example, consider a blind signature scheme where the user, on input a message b , always queries $y = \mathcal{O}(b)$ and includes y as part of the signature; verification checks whether $\mathcal{O}(b) = y$ (among other things). In such a case the query $\mathcal{O}(1)$ may not be in the set $\text{Find}_\delta(\text{trans}_0)$.

As in [BMG07], we handle this issue by introducing a phase in which the forger makes any “heavy” queries that are made by the user with high probability. If the forger knows the correct answers to all these high-probability queries then it is very unlikely that it will incorrectly answer some query asked during the verification of the forged signature.

Given this intuition we now present the details of the attack. The main structure of the attack is based on [BMG07] with necessary changes to adapt the proof to our setting. In particular, our attack makes only polynomially many oracle queries (regardless of the number of queries the scheme itself makes).

Theorem 5.4.3. *Let BS be an oracle blind signature scheme (with perfect completeness) where each party has access to a random oracle. Let $q = \text{poly}(n)$ be an upper bound on the number of oracle queries made by KG, S, U, and Vf. Then there exists an adversary which makes at most $\text{poly}(n)$ queries and breaks the unforgeability of the scheme with non-negligible probability, where the probability is taken over the randomness of the oracle, key generation, and the randomness of the adversary.*

Proof. To prove this theorem we first describe the attacker and analyze its success probability afterwards.

THE ATTACKER. Our adversary \mathcal{U}^* works as follows:

SETUP: KEY GENERATION. The input of the attacker \mathcal{U}^* is a public-key pk . It picks random values $r_0 \leftarrow \{0, 1\}^n$ and $r_1 \leftarrow \{0, 1\}^n$.

STEP 1: REQUESTING A SIGNATURE. The adversary \mathcal{U}^* engages in an interactive signing protocol with the external signing oracle on the message 0. \mathcal{U}^* executes the honest user algorithm $\mathcal{U}(pk, 0; r_0)$ obtaining a valid signature σ_0 . \mathcal{U}^* then verifies the received signature (observing the oracle queries made by the verification). Let trans_0 be the transcript (not including the randomness r_0) of this protocol execution. \mathcal{U}^* then computes the set $\mathcal{I}_0 \leftarrow \text{Find}_\delta(\text{trans}_0)$. Remember, that by the properties of Find_δ the set \mathcal{I}_0 contains all the intersection queries made by the signer (including key generations) and user (including verification) for this signature. Next, denote by T_0 the *complete* transcript of the algorithm run so far. I.e., we assume that T_0 contains the secret key sk , public key pk , the message-signature pair $(0, \sigma_0)$, the randomness r_0 of the user and all query-answer pairs made by the key generation $Q(\text{KG})$, the signer $Q(\text{S})$, and the user $Q(\text{U})$. Note that since the user verifies the generated signature, the set $Q(\text{U})$ includes the queries asked by verification $Q(\text{Vf}(0))$. Note further that the attacker \mathcal{U}^* has only partial knowledge of T_0 .

STEP 2: LEARNING QUERY/ANSWER PAIRS. Let L_0 be the information that \mathcal{U}^* has about T_0 and the oracle \mathcal{O} following Step 1. This includes $pk, 0, r_0, \sigma_0, Q(\mathcal{U}_0)$ and \mathcal{I}_0 . Let q be an upper bound on the number of queries asked by each of the BS protocols and let $\delta = 1/10$ be the failure probability of the algorithm Find_δ . Let $\epsilon = \delta/q$ and $M = q/\epsilon\delta = 100q^2$. For $i = 1, \dots, M$ do the following:

1. Let D_{i-1} be the distribution of T_0 , the transcript of the first step, conditioned on only knowing L_{i-1} .
2. Denote by $Q(L_{i-1})$ the oracle queries that appear in L_{i-1} . If a query $x \in \{0, 1\}^\ell \setminus Q(L_{i-1})$ appears with probability at least ϵ in D_{i-1} , then \mathcal{U}^* makes this query to \mathcal{O} and adds the query/answer pair to L_i . If there is more than one such query, then he adds the lexicographically first one.

STEP 3: SAMPLING A POSSIBLE TRANSCRIPT. \mathcal{U}^* samples a random transcript \tilde{T}_0 according to the distribution D_M . Observe that \tilde{T}_0 also defines a secret key \tilde{sk} that may be distinct from the real secret key sk . Moreover, \tilde{T}_0 may include some new mappings that were not defined in L_M . These most likely will not match the real oracle \mathcal{O} . We let $\tilde{\mathcal{O}}$ be the following oracle. If a query x appears in \tilde{T}_0 then $\tilde{\mathcal{O}}(x)$ returns the value contained in \tilde{T}_0 . Otherwise, $\tilde{\mathcal{O}}(x) = \mathcal{O}(x)$.

STEP 4: FORGING. To forge a signature, \mathcal{U}^* runs the interactive signing protocol locally using \tilde{sk} and $\tilde{\mathcal{O}}$, i.e., $\sigma_1 \leftarrow \langle \mathcal{S}^{\tilde{\mathcal{O}}}(\tilde{sk}), \mathcal{U}^{\tilde{\mathcal{O}}}(pk, 1; r_1) \rangle$ on the message 1. It then verifies the signature σ_1 for the message 1 using the real oracle \mathcal{O} . If the signature verifies, then \mathcal{U}^* outputs (σ_0, σ_1) and aborts otherwise.

THE ANALYSIS. \mathcal{U}^* makes at most $\text{poly}(n) = M + 10^4 q^2 / \delta^2 + O(q)$ oracle queries: M for the learning queries step, $10^4 q^2 / \delta^2$ for running Find_δ , and $O(q)$ for generating and verifying the two signatures. We will argue that \mathcal{U}^* outputs a successful forgery with probability at least $4/5 - \delta - \text{negl}(n)$. To analyze the success probability of \mathcal{U}^* let $\tilde{Q}(\text{KG}), \tilde{Q}(\mathcal{S})$ be the queries made by the key generation and the signer during the computation of σ_1 . We denote by $\tilde{Q}(\mathcal{U})$ the user's queries to $\tilde{\mathcal{O}}$ during the computation of σ_1 . Note that our forger only initiated a single protocol execution with the signer but returns two message/signature pairs. Since the forger runs the honest user protocol in the first execution, $(0, \sigma_0)$ is a valid message/signature pair. Thus, \mathcal{U}^* wins in the unforgeability game as long as $\text{Vf}^{\mathcal{O}}(pk, 1, \sigma_1) = 1$.

In the following we show that, with high probability, the verification algorithm on $(1, \sigma_1)$ never asks a query on which the oracles $\tilde{\mathcal{O}}$ and \mathcal{O} disagree. But if the verification algorithm does not ask such a query, it follows by the perfect completeness of the signature scheme that $(1, \sigma_1)$ must verify as well.

Lemma 5.4.4. *Let $Q(\text{Vf}_1)$ denote the set of oracle queries made when verifying the signature σ_1 . Let $\tilde{Q}(\text{KG})$ and $\tilde{Q}(\mathcal{S}_0)$ denote the set of oracle queries made by the key-generation and signing algorithms, respectively, in the sampled transcript \tilde{T}_0 . Then with*

probability at least $\frac{4}{5} - \delta - \text{negl}(n)$ it holds that

$$Q(\text{Vf}(1)) \cap \left(\tilde{Q}(\text{KG}) \cup \tilde{Q}(\mathcal{S}_0) \right) \subseteq \text{Find}_\delta(\text{trans}_0).$$

Lemma 5.4.4 implies Theorem 5.4.3. To see this, note that $\text{Vf}^{\tilde{\mathcal{O}}}(pk, 1, \sigma_1) = 1$ by perfect completeness of the signature scheme. But the only queries on which $\tilde{\mathcal{O}}$ and \mathcal{O} can possibly differ are queries in $\left(\tilde{Q}(\text{KG}) \cup \tilde{Q}(\mathcal{S}_0) \right) \setminus \text{Find}_\delta(\text{trans}_0)$. If verification makes no such queries, then

$$\text{Vf}^{\mathcal{O}}(pk, 1, \sigma_1) = \text{Vf}^{\tilde{\mathcal{O}}}(pk, 1, \sigma_1) = 1.$$

Let E denote the event considered in Lemma 5.4.4. The proof of Lemma 5.4.4 follows the proof in [BMG07]: we define a series of hybrid distributions where the first hybrid corresponds to the invented transcript \tilde{T}_0 and the transcript of the forger's signature and verification protocols for 1 and the last hybrid corresponds to the transcript produced if all these procedures are executed with respect to the real oracle \mathcal{O} . The biggest difference between our proof and [BMG07] (among the other aspects mentioned in Section 5.2.1) is in the proof of Claim 5.4.5, where we analyze the probability that E happens in the last hybrid. We show that due to the blindness of the signature scheme, event E holds for any pair of messages, and in particular the fixed messages 0 and 1. On the other hand, [BMG07] are only able to show that it is possible to find two messages for which event E occurs by searching through exponentially many (in the number of oracle queries made by the signature scheme) messages. This difference allows our attack to be much more efficient than theirs and in particular, our attack only needs polynomially many oracle queries.

Definition of Hybrid Distributions. We formally define four hybrid distributions \mathcal{H}^0 , \mathcal{H}^1 , \mathcal{H}^2 , and \mathcal{H}^3 as follows:

- \mathcal{H}^0 . The first hybrid is the distribution (\tilde{T}_0, T_1) , where \tilde{T}_0 is the invented transcript created by \mathcal{U}^* in Step 3 and T_1 is the transcript of the signature issue and verification protocols for 1 in Step 4. Note that \tilde{T}_0 also includes the queries of the key generation, while T_1 does not.
- \mathcal{H}^1 . The second hybrid is defined identically to \mathcal{H}^0 , except that we use $\tilde{\mathcal{O}}$ to verify the forger's signature σ_1 . In \mathcal{H}^0 , the \mathcal{O} oracle is used instead.
- \mathcal{H}^2 . The third hybrid has the same distribution as \mathcal{H}^1 , except that we change the definition of $\tilde{\mathcal{O}}$ as follows. Recall that L_M is the set of \mathcal{O} query/answer pairs that \mathcal{U}^* knows after the learning queries step (Step 2). We define $\tilde{\mathcal{O}}$ to answer any query contained in L_M with the answer stored there and all other queries with a

randomly chosen value. This modification results in an oracle $\tilde{\mathcal{O}}$ that agrees with \mathcal{O} on all the queries \mathcal{U}^* has asked from \mathcal{O} until the end of Step 2 and all the other queries are answered completely at random.

\mathcal{H}^3 . The distribution of the last hybrid is the same as \mathcal{H}^2 except that \tilde{T}_0 is replaced with T_0 . Thus the output of this hybrid is (T_0, T_1) which describes the experiment where (1) the keys are generated $(sk, pk) \leftarrow \text{KG}$, (2) the signing algorithm uses sk to run $\sigma_0 \leftarrow \langle \mathcal{S}^{\mathcal{O}}(sk), \mathcal{U}^{\mathcal{O}}(pk, 0; r_0) \rangle$ and $\sigma_1 \leftarrow \langle \mathcal{S}^{\mathcal{O}}(sk), \mathcal{U}^{\mathcal{O}}(pk, 1; r_1) \rangle$, and (3) the verification algorithm uses pk to verify both signatures. Note that all algorithms here use the “real” oracle \mathcal{O} and thus verification succeeds for both signatures.

The distributions considered in each hybrid are taken over random choice of the oracle and random coins of the key-generation algorithm, the signer, and the adversary. We prove Lemma 5.4.4 by showing that (1) event E occurs with high probability in \mathcal{H}^3 and (2) the probability that event E occurs in \mathcal{H}^0 is not much smaller than its probability in \mathcal{H}^3 .

We first show that E occurs with high probability in \mathcal{H}^3 . The following is an immediate consequence of Lemma 5.4.2.

Claim 5.4.5. $\Pr_{\mathcal{H}^3}[E] \geq 1 - \delta - \text{negl}(n)$.

We next show that the probability of E remains unchanged when we move from \mathcal{H}^3 to \mathcal{H}^2 .

Claim 5.4.6. $\mathcal{H}^2 \equiv \mathcal{H}^3$. Thus, $\Pr_{\mathcal{H}^2}[E] = \Pr_{\mathcal{H}^3}[E]$.

Proof. The proof here is the same as in [BMG07]. We can view \mathcal{H}^3 as being sampled as follows: first, fix L_M ; then choose the transcript T_0 at random from \mathcal{D}_M . This, however, is exactly the same distribution as \mathcal{H}^2 where L_M is fixed and we then choose \tilde{T}_0 from \mathcal{D}_M . \square

For the next claim, we need the following definition.

Definition 5.4.7 (Statistical distance). If X, Y are two random variables taking values in a finite set A , then $\text{SD}(X, Y) = 1/2 \cdot \sum_{a \in A} |\Pr[X = a] - \Pr[Y = a]|$.

We now show that \mathcal{H}^1 and \mathcal{H}^2 are “close”.

Claim 5.4.8. $\text{SD}(\mathcal{H}^1, \mathcal{H}^2) \leq \frac{1}{5}$. Thus, $\Pr_{\mathcal{H}^1}[E] \geq \Pr_{\mathcal{H}^2}[E] - \frac{1}{5}$.

Proof. Let $Q(T_0)$ be the queries contained in the transcript T_0 . Let B be the event that \mathcal{U}^* ever asks a query in $Q(T_0) \setminus Q(L_M)$. It is clear that $\mathcal{H}^1 = \mathcal{H}^2$ as long as event B does not occur in either of them, since in both distributions any queries outside of $Q(T_0)$ are answered randomly. This implies that $\Pr_{\mathcal{H}^1}[B] = \Pr_{\mathcal{H}^2}[B]$, and $\text{SD}(\mathcal{H}^1, \mathcal{H}^2) \leq \Pr_{\mathcal{H}^2}[B]$. We now show that $\Pr_{\mathcal{H}^2}[B] \leq \frac{1}{5}$. (In the following, all probabilities are in \mathcal{H}^2 .)

Recall that in Step 2 of the attack, we set $\epsilon = \delta/q$ and \mathcal{U}^* learns at most $M = 100q^2$ query/answer pairs from \mathcal{O} . Let D_i be the distribution of T_0 sampled in this step by \mathcal{U}^* given the set L_i of known query/answer pairs. Let C be the event that there are more than M queries that become likely during the attack. That is, C is the event that there exists a query $x \notin Q(L_M)$ such that x is asked in D_M with probability at least ϵ . Below, we show that $\text{Prob}[C] \leq \delta = \frac{1}{10}$ and $\Pr[B \mid \neg C] \leq \delta = \frac{1}{10}$. This completes the proof, since then

$$\begin{aligned} \text{Prob}[B] &= \text{Prob}[C] \cdot \Pr[B \mid C] + \text{Prob}[\neg C] \cdot \Pr[B \mid \neg C] \\ &\leq \text{Prob}[\neg C] + \Pr[B \mid \neg C] \leq 2\delta = \frac{1}{5}. \end{aligned}$$

The following two claims complete the proof that \mathcal{H}^1 and \mathcal{H}^2 are close.

Claim 5.4.9. *Let C be the event defined in the proof of the previous claim. Then $\Pr_{\mathcal{H}^2}[C] \leq \delta$.*

Proof. All probabilities here are in \mathcal{H}^2 . Consider an arbitrary query x and let hit_x be the event that x is queried to \mathcal{O} by the signer and then by the user when generating the signature on ‘0’. Let $q_x = \Pr[\text{hit}_x]$. Finally, let $A_x(i)$ be the event that x is asked in the i th iteration of Step 2; let $p_x(i) = \Pr[A_x(i)]$; and let $p_x = \Pr[\cup_i A_x(i)]$. Note that $\sum_x q_x \leq q$ since q is an upper bound on the total number of queries asked when running each algorithm of the blind signature scheme. Furthermore, $q_x \geq \epsilon p_x$ because

$$q_x = \text{Prob}[\text{hit}_x] \geq \sum_i \text{Prob}[\text{hit}_x \mid A_x(i)] \cdot \text{Prob}[A_x(i)],$$

and \mathcal{U}^* adds a query to its list only if the probability that this query is asked is at least ϵ . Thus, $\text{Prob}[\text{hit}_x \mid A_x(i)] \geq \epsilon$ and so $q_x \geq \epsilon \sum_i \text{Prob}[A_x(i)] = \epsilon p_x$.

Assume for the sake of contradiction that $\text{Prob}[C] > \delta$. Since C is the event that M queries are learned in Step 2, this implies that the expected number of queries asked, $\sum_x p_x$, is larger than δM . But this would imply

$$\delta M < \sum_x p_x \leq \sum_x q_x / \epsilon \leq q / \epsilon,$$

contradicting the fact that $M = q/\delta\epsilon$. □

Claim 5.4.10. *Let B and C be as defined earlier. Then $\Pr_{\mathcal{H}^2}[B \mid \neg C] \leq \delta$.*

Proof. Recall that in Step 4 \mathcal{U}^* relies only on the mappings stored in L_M , and all queries from $Q(T_0) \setminus Q(L_M)$ are answered at random. But then \mathcal{H}^2 is independent of T_0 conditioned on L_M (whereas L_M has the distribution D_M). This means that we can imagine defining \mathcal{H}^2 by choosing L_M first, then running \mathcal{U}^* (using L_M) to sample \mathcal{H}^2 , and then choosing T_0 conditioned on L_M and \mathcal{H}^2 . Recall that event C is determined by L_M , and assume that L_M is such that event $\neg C$ occurs. This implies that every query asked by \mathcal{U}^* that is not in $Q(L_M)$ must appear in D_M with probability less than ϵ . Since \mathcal{U}^* asks at most q queries in Step 4, the probability that $Q(T_0) \setminus Q(L_M)$ contains one of these queries is at most $\epsilon q = \delta$. \square

Finally, we show that E occurs with the same probability in \mathcal{H}^0 and \mathcal{H}^1 .

Claim 5.4.11. $\Pr_{\mathcal{H}^0}[E] = \Pr_{\mathcal{H}^1}[E]$.

Proof. This claim follows easily if both hybrid distributions \mathcal{H}^0 and \mathcal{H}^1 use the same oracle \mathcal{O} and if they are sampled using the same random coins for key generation and the adversary (note that the randomness of the adversary fully determines the randomness used to run the honest user algorithm during the signature-issue protocol). But then it follows that event E occurs in \mathcal{H}^0 if and only if it also occurs in \mathcal{H}^1 . \square

This completes the proof of Lemma 5.4.4, and thus the proof of Theorem 5.4.3.

5.5 Extensions

5.5.1 Imperfect Completeness

In this section we extend our impossibility result to schemes that are not perfectly complete. That is, we assume that BS is a blind signature scheme that always accepts an honestly generated signature with probability $1 - \text{negl}(n)$.

Definition 5.5.1 (Imperfect Completeness). *An oracle blind signature scheme σ has imperfect completeness if with overwhelming probability in $n \in \mathbb{N}$ the following holds: $(sk, pk) \leftarrow \text{KG}^{\mathcal{O}}(1^n)$, any message $m \in \{0, 1\}$ and any σ output by $\mathcal{U}^{\mathcal{O}}$ in the joint execution of $\mathcal{S}^{\mathcal{O}}(sk)$ and $\mathcal{U}^{\mathcal{O}}(pk, m)$ we have $\text{Vf}^{\mathcal{O}}(pk, m, \sigma) = 1$.*

The following Lemma is identical to Lemma 5.4.2, but it holds also for schemes that have imperfect completeness.

Lemma 5.5.2. *Let BS be an oracle blind signature scheme and consider the blindness game as defined in Definition 5.3.2. Let trans_0 be the transcript corresponding to the signature issue protocol $\langle \mathcal{S}(sk), \mathcal{U}(pk, 0) \rangle$ and let $\mathcal{I}_0 \leftarrow \text{Find}_\delta(\text{trans}_0)$. Then the probability that*

$$Q(\text{Vf}(1)) \cap (Q(\text{KG}) \cup Q(\mathcal{S}_0)) \subseteq \mathcal{I}_0$$

is at least $1 - (\delta + \text{negl}(n))$ (where δ depends on Find_δ and can be arbitrary small).

The proof of this lemma is essentially the same as the one of Lemma 5.4.2, but it differs in the case that the attacker \mathcal{S}^* receives (\perp, \perp) instead of two signatures. This case could not happen before as it never occurs in a scheme with perfect completeness. If the algorithm \mathcal{S}^* gets such a pair, then it simply outputs a random bit. The analysis then follows in a straightforward way because this event happens only with negligible probability.

The forgery attack then proceeds just as in the case for perfect completeness. Since the probability that one of the signatures fails is negligible, this possibility only affects the forgery probability a negligible amount and thus we have the following theorem.

Theorem 5.5.3. *Let BS be an oracle blind signature scheme that has imperfect completeness where each party has access to a random oracle. Let $q = \text{poly}(n)$ be an upper bound on the number of oracle queries made by KG, \mathcal{S} , \mathcal{U} , and Vf. Then there exists an adversary which makes at most $\text{poly}(n)$ queries and breaks the unforgeability of the scheme with non-negligible probability, where the probability is taken over the randomness of the oracle, key generation, verification and the randomness of the adversary.*

5.5.2 One-Way Permutations

We now show how to extend our impossibility result to also rule out constructions from one-way permutations. We let \mathcal{O} be a *random permutation oracle* that on input $x \in \{0, 1\}^n$ returns a value $f(x)$ where f is a random permutation over $\{0, 1\}^n$. Since using techniques from [IR89], the Find_δ algorithm of [BMG09] can be modified to work in the random permutation model (with a polynomial blow-up in the number of queries) the blindness attack remains as before. For the forgery attack we prove the following modified version of Theorem 5.4.3.

Theorem 5.5.4. *Let BS be an oracle blind signature scheme for the message space $\{0, 1\}$ in the random permutation oracle. Let $q = \text{poly}(n)$ be an upper bound on the number of oracle queries made by KG, \mathcal{S} , \mathcal{U} , and Vf. Then there exists an adversary which makes at most $\text{poly}(n)$ queries and breaks the unforgeability of the scheme with non-negligible probability, where the probability is taken over the randomness of the oracle, key generation, and the randomness of the adversary.*

The proof of this theorem is analogous to the proof of Theorem 5.4.3 except for the following modifications [BMG07]. Let N denote the number of oracle queries made by the forger in Theorem 5.4.3. We change the random permutation oracle \mathcal{O} into an oracle \mathcal{O}' defined as follows. When \mathcal{O}' is queried on a value x , if $|x| \geq 2 \log N$ then we answer with $\mathcal{O}(x)$, otherwise we answer with a concatenation of all the values $\mathcal{O}(y)$ for all $y \in \{0, 1\}^{|x|}$, i.e. if $|x| = n$, $\mathcal{O}'(x) = \mathcal{O}(0^n) \parallel \mathcal{O}(0^{n-1}1) \parallel \dots \parallel \mathcal{O}(1^n)$. Given this modification we denote by $\text{BS}' = (\text{KG}', \langle \mathcal{S}', \mathcal{U}' \rangle, \text{Vf}')$ the blind signature scheme using the modified oracle. The following lemma implies Theorem 5.5.4.

Lemma 5.5.5. *Given the scheme $\text{BS}' = (\text{KG}', \langle \mathcal{S}', \mathcal{U}' \rangle, \text{Vf}')$, there exists an adversary \mathcal{U}^* that breaks the scheme with probability $\frac{4}{5} - O(1/\text{poly}(q))$.*

Proof. Lemma 5.5.5 implies Theorem 5.5.4 because we can simulate \mathcal{O}' for \mathcal{U}^* given access to \mathcal{O} . If \mathcal{U}^* makes at most N' queries, this simulation makes at most $N' \cdot N^2$ queries to \mathcal{O} .

\mathcal{U}^* is identical to the forger presented in the proof of Theorem 5.4.3 except that we need to deal with the case where we learn a query in \mathcal{O} that outputs the same value as an invented query in \tilde{T}_0 , thus contradicting the permutation property. Fortunately, we show that this only happens with small probability so it does not affect our attack by much.

To prove this lemma we define the same hybrid distributions $\mathcal{H}^0, \mathcal{H}^1, \mathcal{H}^2$, and \mathcal{H}^3 but now we have to show that these distributions are closely distributed even once we modify the attack. The only place where the proof changes is in comparing \mathcal{H}^1 and \mathcal{H}^2 . We prove that (analogously to Claim 5.4.8) $\text{SD}(\mathcal{H}^1, \mathcal{H}^2) = \frac{1}{5} + O(1/\text{poly}(q))$. Recall that these two hybrids differ only if \mathcal{U}^* queries a value $x \in Q(T_0) \setminus Q(L_M)$. Following the analysis of Claim 5.4.8, such a query occurs with probability at most $1/5$ (in both experiments). Now, assume that \mathcal{U}^* does not make such a query.

In \mathcal{H}^1 when a query is asked during signing or verification it may output the same answer as a guessed answer for a query x' in \tilde{T}_0 (event hit_1). On the other hand, in \mathcal{H}^2 the answer to a query (again, during signing or verification) x is never equal to a query \tilde{T}_0 but it might be equal to one in $Q(T_0) \setminus Q(L_M)$ (event hit_2).

Observe that $(\mathcal{H}^1 | \neg \text{hit}_1) \equiv (\mathcal{H}^2 | \neg \text{hit}_2)$. Thus, to complete the proof, we show that $\text{Prob}[\text{hit}_i] = O(1/\text{poly}(q))$ for $i = 1, 2$ and therefore $\text{SD}(\mathcal{H}^1, \mathcal{H}^2) = \frac{1}{5} + O(1/\text{poly}(q))$. This follows from the fact that \mathcal{U}^* makes N' queries and thus the answer to each query is chosen from a set of size at least $N^2 - N'$. Thus, this query hits a previous answer with probability at most $N'/(N^2 - N') = O(1/\text{poly}(q))$. \square

Bibliography

- [Abe01a] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *Advances in Cryptology — Eurocrypt'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151. Springer-Verlag, 2001.
- [Abe01b] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 136–151, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.
- [AFG⁺10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In *Advances in Cryptology – CRYPTO 2010*, Lecture Notes in Computer Science, pages 209–236, Santa Barbara, CA, USA, August 2010. Springer, Berlin, Germany.
- [AO09] Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 435–450, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.
- [ASW98] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology — Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 1998.
- [Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS) 2001*, pages 106–115. IEEE Computer Society Press, 2001.
- [BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC)2002*, pages 484–493. ACM Press, 2002.

- [BMG07] Boaz Barak and Mohammad Mahmoody-Ghidary. Lower bounds on signatures from symmetric primitives. In *48th Annual Symposium on Foundations of Computer Science*, pages 680–688, Providence, USA, October 20–23, 2007. IEEE Computer Society Press.
- [BMG09] 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 Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 374–390, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.
- [BMV08a] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In *Topics in Cryptology – Cryptographer’s Track, RSA Conference (CT-RSA) ’08*, volume 4964 of *Lecture Notes in Computer Science*, pages 71–87. Springer-Verlag, 2008.
- [BMV08b] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the “one-more” computational problems. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 71–87, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.
- [BNPS03a] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum’s blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.
- [BNPS03b] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum’s blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
- [Bol03a] Alexandra Boldyreva. Efficient threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public-Key Cryptography (PKC) ’03*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer-Verlag, 2003.
- [Bol03b] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46, Miami, USA, January 6–8, 2003. Springer, Berlin, Germany.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS*

93: *1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

- [Bro06] Daniel Brown. What hashes make RSA-OAEP secure? Number 2006/223, 2006.
- [Bro07] Daniel Brown. Irreducibility to the one-more evaluation problems: More may be less. Number 2007/435, 2007.
- [BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT’98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Germany.
- [Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13:143–202, 2000.
- [CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2000*, pages 235–244. ACM Press, 2000.
- [CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.
- [Cha83a] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO’82*, pages 199–203, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.
- [Cha83b] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology — Crypto’82*, pages 199–203. Plenum, New York, 1983.
- [Cha84] David Chaum. Blind signature system. In David Chaum, editor, *Advances in Cryptology – CRYPTO’83*, page 153, Santa Barbara, CA, USA, 1984. Plenum Press, New York, USA.
- [CHL06] Yan-Cheng Chang, Chun-Yuan Hsiao, and Chi-Jen Lu. The impossibility of basing one-way permutations on central cryptographic primitives. *Journal of Cryptology*, 19(1):97–114, January 2006.
- [CKW04a] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In *Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148. Springer-Verlag, 2004.

- [CKW04b] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04: 4th International Conference on Security in Communication Networks*, volume 3352 of *Lecture Notes in Computer Science*, pages 134–148, Amalfi, Italy, September 8–10, 2004. Springer, Berlin, Germany.
- [CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. Simulatable adaptive oblivious transfer. In *Advances in Cryptology — Eurocrypt’07*, Lecture Notes in Computer Science, pages 573–590. Springer-Verlag, 2007.
- [Cor02a] Jean-Sebastien Coron. Optimal security proofs for pss and other signature schemes. In *Advances in Cryptology — Eurocrypt2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer-Verlag, 2002.
- [Cor02b] Jean-Sébastien Coron. Security proof for partial-domain hash signature schemes. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 613–626, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.
- [DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.
- [DPP97] Ivan Damgård, Torben Pedersen, and Birgit Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. volume 10, pages 163–194. Springer-Verlag, 1997.
- [Fis02] Marc Fischlin. On the impossibility of constructing non-interactive statistically-secret protocols from any trapdoor one-way function. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 79–95, San Jose, CA, USA, February 18–22, 2002. Springer, Berlin, Germany.
- [Fis06a] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *Advances in Cryptology — Crypto’06*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77. Springer-Verlag, 2006.
- [Fis06b] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.
- [FS09] Marc Fischlin and Dominique Schröder. Security of blind signatures under aborts. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th*

International Conference on Theory and Practice of Public Key Cryptography, volume 5443 of *Lecture Notes in Computer Science*, pages 297–316, Irvine, CA, USA, March 18–20, 2009. Springer, Berlin, Germany.

- [FS10] Dario Fiore and Dominique Schröder. Uniqueness is a different story: Impossibility of verifiable random functions from trapdoor permutations. *Cryptology ePrint Archive*, 2010. <http://eprint.iacr.org/>.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33:792–807, 1986.
- [GK96] Oded Goldreich and Hugo Krawczyk. On the composition of Zero-Knowledge Proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.
- [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*, pages 325–335, Redondo Beach, California, USA, November 12–14, 2000. IEEE Computer Society Press.
- [GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, USA, May 15–17, 1989. ACM Press.
- [GMPY06] Juan Garay, Philip MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. In *Theory of Cryptography Conference (TCC)'06*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer-Verlag, 2006.
- [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*, pages 126–135, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press.
- [GO93] Shafi Goldwasser and Rafail Ostrovsky. Invariant signatures and non-interactive zero-knowledge proofs are equivalent (extended abstract). In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 228–245, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Germany.
- [Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, 34(6):277–281, 1990.
- [Gol04] Oded Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

- [GT00] Rosario Gennaro and Luca Trevisan. Lower bounds on the efficiency of generic cryptographic constructions. In *41st Annual Symposium on Foundations of Computer Science*, pages 305–313, Redondo Beach, California, USA, November 12–14, 2000. IEEE Computer Society Press.
- [HILL99a] Johan Håstad, Russel Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [HILL99b] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.
- [HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 111–129, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany.
- [HKKL07a] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *Theory of Cryptography Conference (TCC) '07*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341. Springer-Verlag, 2007.
- [HKKL07b] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Berlin, Germany.
- [HRV10] Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency improvements in constructing pseudorandom generators from one-way functions. In Leonard J. Schulman, editor, *42nd Annual ACM Symposium on Theory of Computing*, pages 437–446, Cambridge, Massachusetts, USA, June 5–8, 2010. ACM Press.
- [IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity-based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press.
- [ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *21st Annual ACM Symposium on*

Theory of Computing, pages 12–24, Seattle, Washington, USA, May 15–17, 1989. ACM Press.

- [IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, Washington, USA, May 15–17, 1989. ACM Press.
- [JLO97a] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology — Crypto’97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 1997.
- [JLO97b] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO’97*, volume 1294 of *Lecture Notes in Computer Science*, pages 150–164, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.
- [KST99] Jeong Han Kim, Daniel R. Simon, and Prasad Tetali. Limits on the efficiency of one-way permutation-based hash functions. In *40th Annual Symposium on Foundations of Computer Science*, pages 535–542, New York, New York, USA, October 17–19, 1999. IEEE Computer Society Press.
- [KZ05] Aggelos Kiayias and Hong-Sheng Zhou. Two-round concurrent blind signatures without random oracles. Number 2005/435 in Cryptology eprint archive. eprint.iacr.org, 2005.
- [KZ08] Aggelos Kiayias and Hong-Sheng Zhou. Equivocal blind signatures and adaptive UC-security. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 340–355, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.
- [Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
- [Lin03] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC)2003*, pages 683–692. ACM Press, 2003.
- [LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78, Barcelona, Spain, May 20–24, 2007. Springer, Berlin, Germany.

- [Lub92] Michael Luby. Pseudo-random generators from one-way functions (abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO’91*, volume 576 of *Lecture Notes in Computer Science*, page 300, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Berlin, Germany.
- [Nao91] Moni Naor. Bit commitment using pseudo-randomness. *Journal of Cryptology*, 4(2):151–158, 1991.
- [NP05] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, January 2005.
- [NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, Washington, USA, May 15–17, 1989. ACM Press.
- [Oka06a] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In *Theory of Cryptography Conference (TCC)’06*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99. Springer-Verlag, 2006.
- [Oka06b] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 80–99, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.
- [PS96] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology – ASIACRYPT’96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265, Kyongju, Korea, November 3–7, 1996. Springer, Berlin, Germany.
- [PS00a] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [PS00b] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
- [PV06a] Pascal Paillier and Jorge L. Villar. Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 252–266, Shanghai, China, December 3–7, 2006. Springer, Berlin, Germany.

- [PV06b] Pascal Paillier and Jorge Luis Villar. Trading one-wayness against chosen-ciphertext security in factoring-based encryption. In *Advances in Cryptology — Asiacrypt'06*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
- [Rab81] Michael Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, 1981.
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.
- [RTV04a] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.
- [RTV04b] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography Conference (TCC)2004*, Lecture Notes in Computer Science, pages 1–20. Springer-Verlag, 2004.
- [Rud92] Steven Rudich. The use of interaction in public cryptosystems (extended abstract). In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 242–251, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Berlin, Germany.
- [Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Germany.