On the Impossibility of Three-Move Blind Signature Schemes

Marc Fischlin Dominique Schröder

Darmstadt University of Technology, Germany
www.minicrypt.de

Abstract. We investigate the possibility to prove security of the well-known blind signature schemes by Chaum, and by Pointcheval and Stern in the standard model, i.e., without random oracles. We subsume these schemes under a more general class of blind signature schemes and show that finding security proofs for these schemes via black-box reductions in the standard model is hard. Technically, our result deploys meta-reduction techniques showing that black-box reductions for such schemes could be turned into efficient solvers for hard non-interactive cryptographic problems like RSA or discrete-log. Our technique yields significantly stronger impossibility results than previous meta-reductions in other settings by playing off the two security requirements of the blind signatures (unforgeability and blindness).

1 Introduction

Blind signatures [Cha83] implement a carbon copy envelope allowing a signer to issue signatures for messages such that the signer's signature on the envelope is imprinted onto the message in the sealed envelope. In particular, the signer remains oblivious about the message (blindness), but at the same time no additional signatures without the help of the signer can be created (unforgeability).

Many blind signature schemes have been proposed in the literature, e.g., [Cha83, JLO97, PS00, Abe01, Bol03, CKW04, KZ06, Fis06, Oka06, HK07, HKKL07, FS09, AO09], with varying security and efficiency characteristics. The arguably most prominent examples are the schemes by Chaum [Cha83] based on RSA and the ones by Pointcheval and Stern [PS00] 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 [BNPS03].

Here 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., where 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.

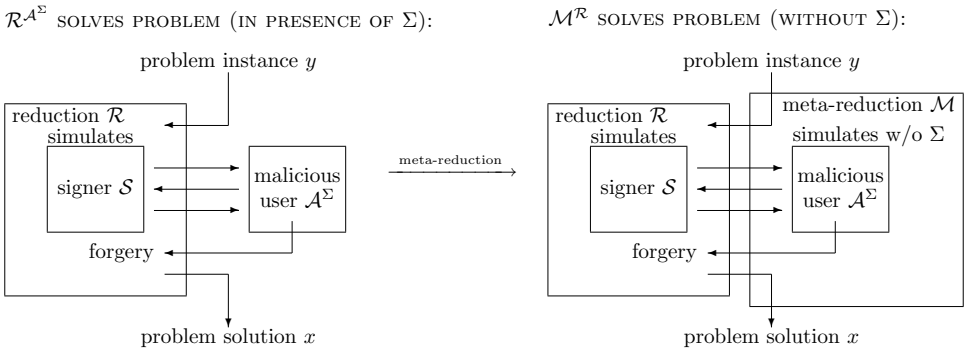

Figure 1: Meta-reduction technique: The black-box reduction $\mathcal{R}$ on the left hand side uses the adversary $\mathcal{A}^\Sigma$ against unforgeability to solve an instance y of the non-interactive problem. The meta-reduction $\mathcal{M}$ on the right hand side then uses $\mathcal{R}$ to solve the problem from scratch, i.e., by simulating $\mathcal{A}^\Sigma$ without Σ. For this, the meta-reduction $\mathcal{M}$ exploits the blindness property of the scheme.

We note that the construction by Boldyreva [Bol03] 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 [CNS07, FS09] for further discussions).

1.1 The Idea Behind our Result

Given a blind signature scheme with the properties 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 or 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 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 infeasbile, 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 $\mathcal{A}^\Sigma$ breaking the unforgeability of the blind signature scheme with the help of Σ. By assumption, the reduction $\mathcal{R}$ with access to $\mathcal{A}^\Sigma$ is then able to break the underlying cryptographic problem (see the left part of Figure 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,Cor02,Bro06,PV06], 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 1). To this end, the meta-reduction $\mathcal{M}$ fills in for adversary $\mathcal{A}^\Sigma$ and simulates the adversary's actions without Σ, mainly by resetting the reduction $\mathcal{R}$ appropriately. We have then 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 $\mathcal{A}^{\Sigma}$, 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 $\mathcal{A}^{\Sigma}$ 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.

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, RTV04]) 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, Cor02, Bro06, PV06, Bro07, BMV08] 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:

- 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 [PV06, Bro07, BMV08] 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 [PV06] 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.

[1] We 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.

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 underyling 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.

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 [Fis06], but there are also examples where blindness and unforgeability are correlated, as in Abe's scheme [Abe01] 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 sent 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 [Oka06] and by Fischlin [Fis06] 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.

1.4 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 [Oka06] based on the 2SDH bilinear Diffie-Hellman assumption. Any schemes with three moves or less either use the random oracle model [Cha83, PS00, Bol03] or a commom reference string [Fis06, HK07, AO09].

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 [HKKL07]. 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. [BMV08] show meta-reduction based separations of the one-more RSA and one-more discrete-logarithm problem from their regular counterparts. The conclusion in [BMV08] 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, BMV08] 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 [BMV08]). 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.

Organization. We start with the definition of blind signature schemes in Section 2. In Section 3 we discuss our notion of black-box reductions to hard problems. Before presenting our main result in Section 5 where we show the hardness of finding black-box reductions from unforgeability to non-interactive problems we first discuss a simpler case for restricted reductions in Section 4 to provide some intuition about the general result. We have delegated the case of computational blindness to Appendix A.

2 Blind Signatures

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

Key Generation. $\mathsf{KG}(1^n)$ *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. $\mathsf{Vf}(pk, m, \sigma)$ *outputs a bit.*

It is assumed that the scheme is complete*, i.e., for any* $(sk, pk) \leftarrow \mathsf{KG}(1^k)$*, 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* $\mathsf{Vf}(pk, m, \sigma) = 1$.

Security of blind signature schemes requires two properties, unforgeability and blindness [JLO97,PS00]. 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 2.2 (Secure Blind Signature Scheme) *A blind signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *is called secure if the following holds:*

Unforgeability. *For any efficient algorithm* $\mathcal{U}^*$ *the probability that experiment* $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}$ *evaluates to* 1 *is negligible (as a function of* n*) where*

Experiment $\mathsf{Forge}^{\mathsf{BS}}_{\mathcal{U}^*}$
$(sk, pk) \leftarrow \mathsf{KG}(1^n)$
$((m_1, \sigma_1), \ldots, (m_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{U}^{*\langle \mathcal{S}(sk), \cdot \rangle^\infty}(pk)$
Return 1 *iff*
$m_i \neq m_j$ *for* $1 \leq i < j \leq k+1$, *and*
$\mathsf{Vf}(pk, m_i, \sigma_i) = 1$ *for all* $i = 1, 2, \ldots, k+1$, *and*
at most k *interactions with* $\langle \mathcal{S}(sk), \cdot \rangle^\infty$ *were completed.*

Computational resp. Statistical Blindness. *For any (efficient resp. unbounded) algorithm* $\mathcal{S}^*$ *working in modes* ***find****,* ***issue*** *and* ***guess****, the probability that the following experiment* $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}$ *evaluates to* 1 *is negligibly close to* $1/2$*, where*

Experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}$
$(pk, m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{S}^*(\mathsf{find}, 1^n)$
$b \leftarrow \{0, 1\}$
$\mathsf{st}_{\mathsf{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk, m_b) \rangle^1, \langle \cdot, \mathcal{U}(pk, m_{1-b}) \rangle^1}(\mathsf{issue}, \mathsf{st}_{\mathsf{find}})$
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 $(\sigma_0, \sigma_1) = (\bot, \bot)$ *if* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$
$b^* \leftarrow \mathcal{S}^*(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{st}_{\mathsf{issue}})$
return 1 *iff* $b = b^*$.

We remark that, even if occassionally not mentioned, all algorithms in this paper receive the security parameter 1^n as additional input.

3 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. [RTV04] 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 that 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 3.1 (Hard Non-Interactive Problem) *A non-interactive (cryptographic) problem* $P = (I, V)$ *consists of two efficient algorithms:*

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

Instance Verification $V(x, y)$**.** *The instance verification algorithm takes as input a value* x *as well as an instance* y *of a cryptographic problem, and outputs a decision bit.*

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 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 $P_0, P_1, \ldots, P_k$, and one gets a sequence of reductions $\mathcal{R}_1, \ldots, \mathcal{R}_k$ 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 $i = 1, 2, \ldots, k$ and outputs a randomly chosen instance.

4 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.

4.1 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 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 4.1 (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* $\mathsf{SigDerCheck}^{\mathsf{BS}}_{\mathcal{S}^*,\mathsf{SDCh}}$ *evaluates to* 1 *is negligible, where*

Experiment $\mathsf{SigDerCheck}^{\mathsf{BS}}_{\mathcal{S}^*,\mathsf{SDCh}}$
$(pk, m, \mathsf{st}) \leftarrow \mathcal{S}^*(\mathsf{find}, 1^n)$
$(\perp, \sigma) \leftarrow \langle \mathcal{S}^*(\mathsf{issue}, \mathsf{st}), \mathcal{U}(pk, m) \rangle$
where trans *denotes the communication between* $\mathcal{S}^*$, $\mathcal{U}$
$c \leftarrow \mathsf{SDCh}(pk, \mathsf{trans})$
return 1 *if* $\sigma \neq \perp$ *and* $c = 0$, *or if* $\sigma = \perp$ *but* $c = 1$.

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 4.2 (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* $\mathcal{S}^*_{\mathsf{trans}}$ *the probability that the experiment* $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*_{\mathsf{trans}},\Sigma}(n)$ *evaluates to* 1 *is negligibly close to* $1/2$, *where*

[2] The 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.

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

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

$(pk, \mathsf{st}_1, m_{-1}, m_0) \leftarrow \mathcal{S}^{*,\Sigma}_{\mathsf{trans}}(\mathsf{init}, 1^n)$

$\mathsf{st}_2 \leftarrow \mathcal{S}^{*,\Sigma}_{\mathsf{trans}}{}^{\langle\cdot,\mathcal{U}(pk,m_{-1})\rangle^1,\langle\cdot,\mathcal{U}(pk,m_0)\rangle^1}(\mathsf{issue}, \mathsf{st}_1)$

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* $\sigma_1 \leftarrow \Sigma(pk, \mathsf{trans}_{-1}, m_1)$

set $(\sigma_{-1}, \sigma_0, \sigma_1) = (\perp, \perp, \perp)$ *if* $\sigma_{-1} = \perp$ *or* $\sigma_0 = \perp$ *or* $\sigma_1 = \perp$

$b^* \leftarrow \mathcal{S}^{*,\Sigma}_{\mathsf{trans}}(\mathsf{guess}, \mathsf{st}_2, m_{-1}, \sigma_{-1}, m_b, \sigma_b)$

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 $\mathsf{trans} = (y, z)$ denote the transcript of this communication. Algorithm $\Sigma(pk, \mathsf{trans}, m)$ 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 4.3 (Generic Forgery Oracle) *For a statistically-blind signature scheme* BS *the generic forgery oracle* $\Sigma(pk, \mathsf{trans}, m)$ *performs the following steps:*

enumerate all values r *such that*

the user algorithm $\mathcal{U}(pk, m)$ *for randomness* r *generates the same transcript* trans *when fed with the same signer messages as in* 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 4.4 *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 $\mathcal{S}^*_{\mathsf{trans}}$ in experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ with the generic forgery oracle Σ which outputs $b^* = b$ with non-negligible probability beyond $1/2$. Then we construct an adversarial controlled signer $\mathcal{S}^*_{\mathsf{blind}}$ against the blindness (with oracle access to Σ) as follows. Algorithm $\mathcal{S}^*_{\mathsf{blind}}$ invokes $\mathcal{S}^*_{\mathsf{trans}}(\mathsf{init}, 1^n)$ to get $(pk, \mathsf{st}_1, m_{-1}, m_0)$; it uses its access to Σ to answer any request of $\mathcal{S}^*_{\mathsf{trans}}$ to this oracle. Algorithm $\mathcal{S}^*_{\mathsf{blind}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$. In the case that $\mathcal{S}^*_{\mathsf{blind}}$ obtains two undefined signatures from the blindness experiment, i.e., $(\sigma_{-1}, \sigma_0) =$

[3] Note that blindness for 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.

$(\perp, \perp)$, then $\mathcal{S}^*_{\mathsf{blind}}$ returns $(m_{-1}, \perp, m_0, \perp)$ to $\mathcal{S}^*_{\mathsf{trans}}$. Otherwise, if both executions have been successful, then $\mathcal{S}^*_{\mathsf{blind}}$ executes $\mathcal{S}^*_{\mathsf{trans}}$ in mode guess on input $(\mathsf{st}_2, m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ to obtain a bit b^*, where st_2 is the state returned by $\mathcal{S}^*_{\mathsf{trans}}(\mathsf{st}_1)$ after the interaction with the users. Algorithm $\mathcal{S}^*_{\mathsf{blind}}$ 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 $(pk, \mathsf{st}_1, m_{-1}, m_0)$ output by the transcript adversary $\mathcal{S}^*_{\mathsf{trans}}$ *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 $\mathcal{S}^*_{\mathsf{trans}}$ is deterministic and chooses the bad tuple $(pk, \mathsf{st}_1, m_{-1}, m_0)$ 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 $\mathcal{S}^*_{\mathsf{trans}}$). 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 $\mathcal{S}^*_{\mathsf{blind}}$ 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, $\mathcal{S}^*_{\mathsf{blind}}$ performs an almost perfect simulation from $\mathcal{S}^*_{\mathsf{trans}}$'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, $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ is perfect in this case. But then we can conclude that if $\mathcal{S}^*_{\mathsf{trans}}$ succeeds with non-negligible probability over $1/2$, then $\mathcal{S}^*_{\mathsf{blind}}$ also succeeds with non-negligible probability bounded away from $1/2$. □

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 4.5 *For every statistically blind signature scheme 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 4.6 (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 $\mathsf{Vf}(pk, m'_0, \sigma'_0) = 1$ then let $\sigma'_1 \leftarrow \Sigma(pk, \mathsf{trans}'_0, m'_1)$ 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.

4.2 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 4.7 *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.*

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 $\mathsf{msg}i$ 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 $\mathsf{msg}j$ 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.

Meta-reduction $\mathcal{M}(y)$
let $(pk, \mathsf{st}_{\mathsf{init}}) \leftarrow \mathcal{R}(\mathsf{init}, y)$
let $(\mathrm{msg}_1, \mathsf{st}_{\mathsf{msg1}}) \leftarrow \mathcal{R}(\mathsf{msg1}, \mathsf{st}_{\mathsf{init}})$

choose $m_0 \leftarrow \{0,1\}^n$	choose $m_1 \leftarrow \{0,1\}^n$
let $(\mathrm{msg2}_0, \mathsf{st}^0_{\mathrm{msg2}}) \leftarrow \mathcal{U}(\mathsf{msg2}, pk, m_0, \mathrm{msg1})$	let $(\mathrm{msg2}_1, \mathsf{st}^1_{\mathrm{msg2}}) \leftarrow \mathcal{U}(\mathsf{msg2}, pk, m_1, \mathrm{msg1})$
let $(\mathrm{msg3}_0, \mathsf{st}^0_{\mathrm{msg3}}) \leftarrow \mathcal{R}(\mathsf{msg3}, \mathsf{st}_{\mathrm{msg1}}, \mathrm{msg2}_0)$	let $(\mathrm{msg3}_1, \mathsf{st}^1_{\mathrm{msg3}}) \leftarrow \mathcal{R}(\mathsf{msg3}, \mathsf{st}_{\mathrm{msg1}}, \mathrm{msg2}_1)$
let $\sigma_0 \leftarrow \mathcal{U}(\mathsf{finish}, \mathsf{st}^0_{\mathrm{msg2}}, \mathrm{msg3}_0)$	let $\sigma_1 \leftarrow \mathcal{U}(\mathsf{finish}, \mathsf{st}^1_{\mathrm{msg2}}, \mathrm{msg3}_1)$

output $x' \leftarrow \mathcal{R}(\mathsf{final}, \mathsf{st}^0_{\mathrm{msg3}}, m_0, \sigma_0, m_1, \sigma_1)$

Figure 2: Meta-Reduction for Vanilla Reduction (three moves), where $\mathsf{trans}_0 = (\mathrm{msg1}, \mathrm{msg2}, \mathrm{msg3})$ denotes the transcript of the first execution.

Description of the Meta-Reduction. The meta-reduction $\mathcal{M}$ works as follows (see Figure 2 for the case of three moves). It gets as input an instance y of the problem. It start to simulate the reduction $\mathcal{R}$ on y to derive a public key pk as well as the first message msg1 on behalf of the signer and a state $\mathsf{st}_{\mathrm{msg1}}$. 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, σ_0), (m_1, σ_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

$$\mathrm{Prob}\left[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1 \;\middle|\; \mathcal{M} \right]$$

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,

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

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

$$\mathrm{Prob}\left[y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{M}}(y) : V(x', y) = 1 \;\middle|\; \mathcal{M} \right] \approx 0.$$

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

Description of Adversary $\mathcal{S}^*_{\mathsf{trans}}$. 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathsf{st}_{\mathsf{msg1}}$. The algorithm $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ receives two (valid) signatures (σ_{-1}, σ_0) and runs the reduction $\mathcal{R}$ in mode final on input $(\mathsf{st}^0_{\mathrm{msg3}}, m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ 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 $\mathcal{S}^*_{\mathsf{trans}}$. 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 $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ resets $\mathcal{R}$ and uses in the second execution the prefix msg1 (obtained during the signature generation of (m_{-1}, σ_{-1})) in experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$. 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 4.5. For simplicity we neglect the small error for Σ returning an invalid signature in the analysis below.

We obtain for the probability that $\mathcal{S}^*_{\mathsf{trans}}$ outputs the right bit $b^* = b$:

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

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 $\mathcal{S}^*_{\mathsf{trans}}$ returns $b^* = 1$ in the case that the reduction $\mathcal{R}$ returns a valid solution x' of y. Hence,

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

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. □

5 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 $i, j = 1, 2, \ldots, q$ 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 $\mathcal{A}_{i,1}, \ldots, \mathcal{A}_{i,q}$ 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.

5.1 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.

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 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.

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

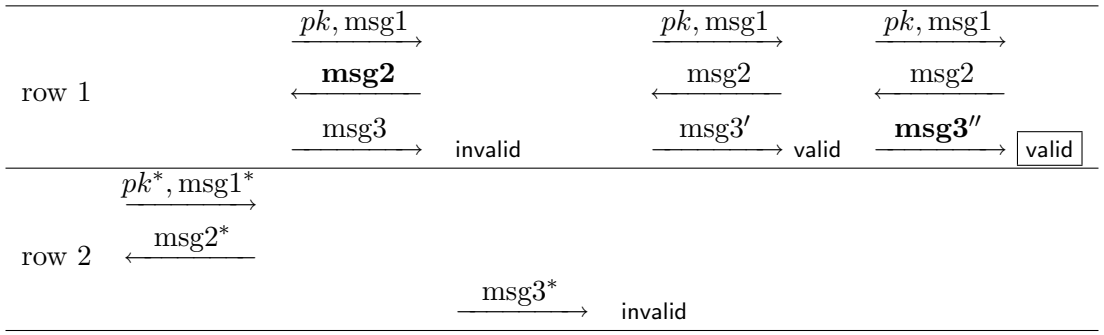

Figure 3: Example of a resetting scheduling with restricted cross-resets (executions in different rows may also run concurrently): Regarding the first and last execution in row 1 there is no other successful execution in between transmission of **msg2** and **msg3″**, except when it uses the same $(pk, \text{msg1})$ 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 $(pk', \text{msg1}')$.

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 function are functions that, when queried for q distinct preimages, output q independently distributed values:

Definition 5.1 (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 $x_1, x_2, \ldots, x_q \in \{0,1\}^a$ and any $y_1, \ldots, y_q \in \{0,1\}^b$ we have*

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

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 $(pk, \text{msg1})$. We note that the length $\psi(n)$ of this randomness is only polynomial by construction of the generic forgery oracle:

Definition 5.2 (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 $\mathsf{Vf}(pk, m_0', \sigma_0') = 1$ then let $\sigma_1' \leftarrow \Sigma^{aug}(pk, \mathsf{trans}_0', m_1'; s_0')$ 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.

5.2 Impossibility Result

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

Theorem 5.3 *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 4.7 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 $pk, \text{msg1}$; 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

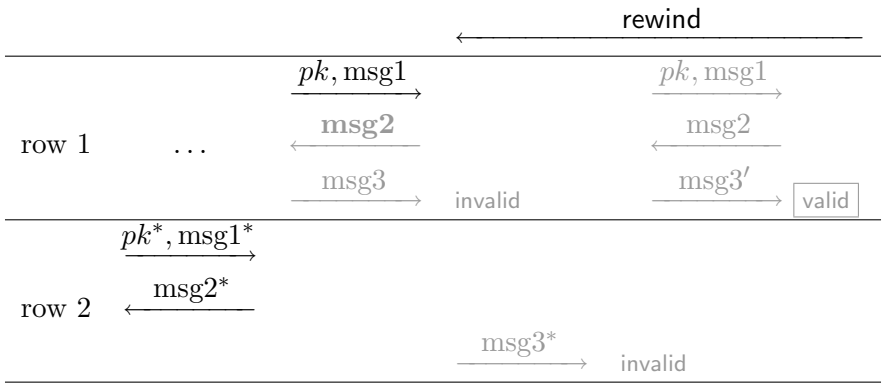

Figure 4: Continuously rewind to first execution in the row in which the same $(pk_{i,j}, \text{msg1}_{i,j})$ has been sent. By the restricted resetting scheduling no other execution can finish successfully meanwhile.

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 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 tanscript-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 wether is has already stored such a pair for the transmission $(pk_{i,j}, \text{msg1}_{i,j})$ is 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-signture 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 $\mathsf{Time}(\mathcal{M})$, 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 $\mathsf{Succ}(Z)$ 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

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

is non-negligible (given $\text{Prob}[\mathsf{Succ}(\mathcal{A})] \not\approx 0$). 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,

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

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 $\mathsf{Bound}_{r(n)}$ denote the event that in each execution $\mathcal{M}$ rewinds $\mathcal{R}$ at most $r(n)$ times. We then divide the success probability $\text{Prob}[\mathsf{Succ}(\mathcal{M})]$ into the

case where $\mathcal{M}$ rewinds the reduction $\mathcal{R}$ more then $r(n)$ 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:

$$\begin{aligned}\mathrm{Prob}[\,\mathsf{Succ}(\mathcal{M})] &\leq \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}) \wedge \mathsf{Bound}_{r(n)}\right] + \mathrm{Prob}\left[\,\neg\mathsf{Bound}_{r(n)}\right] \\ &\leq \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \wedge \mathsf{Bound}_{r(n)}\right] + \mathrm{Prob}\left[\,\neg\mathsf{Bound}_{r(n)}\right] \\ &\leq \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] + \mathrm{Prob}\left[\,\neg\mathsf{Bound}_{r(n)}\right]\end{aligned}$$

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

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

for some polynomial $p(n)$ and infinitely many n's. Let $\mathbb{E}\,[\mathsf{Time}(\mathcal{M})] = t(n)$ 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 $\neg\mathsf{Bound}_{r(n)}$ happens as

$$\mathrm{Prob}\left[\,\neg\mathsf{Bound}_{r(n)}\right] \;\leq\; \mathrm{Prob}[\,\mathsf{Time}(\mathcal{M}) \geq r(n)] \;\leq\; \frac{\mathbb{E}\,[\mathsf{Time}(\mathcal{M})]}{r(n)} \;\leq\; \frac{1}{2p(n)}.$$

Particularly, the probability that $\mathcal{M}$ rewinds the reduction $\mathcal{R}$ more then $r(n)$ 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:

$$\begin{aligned}&\mathrm{Prob}[\,\mathsf{Succ}(\mathcal{A})] - \mathrm{Prob}[\,\mathsf{Succ}(\mathcal{M})] \\ &\qquad\geq \mathrm{Prob}[\,\mathsf{Succ}(\mathcal{A})] - \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] - \mathrm{Prob}\left[\,\neg\mathsf{Bound}_{r(n)}\right] \\ &\qquad\geq \tfrac{1}{2} \cdot \mathrm{Prob}[\,\mathsf{Succ}(\mathcal{A})] - \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] \\ &\qquad\geq \tfrac{1}{2} \cdot \left(\mathrm{Prob}[\,\mathsf{Succ}(\mathcal{A})] - \mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\right).\end{aligned}$$

In the sequel we assume towards contradiction that $\mathrm{Prob}\left[\,\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]$ is negligible. We again use transcript-independence of signatures to derive a contradiction.

Description of Adversary $\mathcal{S}^*_{\mathsf{trans}}$. In order to derive a contradiction we build a successful attacker $\mathcal{S}^*_{\mathsf{trans}}$ against the transcript-independence of signatures. This adversary works similar to the previously described adversary $\mathcal{S}^*_{\mathsf{trans}}$ in the proof of Theorem 4.7. The difference consists in combining the experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ (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 $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ computes an instance $y \leftarrow I(1^n)$ of a cryptographic problem P and selects a random index $k \in \{1, \ldots, q^2\}$. 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ uses its oracle Σ to compute a signature $\sigma^*_{i,j}$ for an independent random message $m^*_{i,j}$. It stores $(i,j,pk_{i,j},\mathrm{msg1}_{i,j},m^*_{i,j},\sigma^*_{i,j})$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ behaves as follows. Let $(pk_{i,j},\mathrm{msg1}_{i,j})$ be the data initially sent by the reduction in this execution. Adversary $\mathcal{S}^*_{\mathsf{trans}}$ executes experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ using $\mathcal{R}$, i.e., $\mathcal{S}^*_{\mathsf{trans}}$ uses the public key sent by $\mathcal{R}$ as his public key during the experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$. It selects two random messages m'_{-1},m'_0 as the challenges and outputs (pk,m'_{-1},m'_0). The adversary $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ stops, outputting a random bit b^*. Otherwise, $\mathcal{S}^*_{\mathsf{trans}}$ proceeds with the simulation as follows. For the other signature generation for m'_0, adversary $\mathcal{S}^*_{\mathsf{trans}}$ guesses how many rewindings (of $\mathcal{R}$) are necessary in order to derive another pair. For this, it selects a random index $t\in\{1,\ldots,r(n)\}$ and computes $t-1$ random messages m^ℓ as well as $t-1$ random strings r^ℓ for $\ell=1,2,\ldots,t-1$. During the ℓ-th repetition for $\ell<t$, adversary $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ aborts and outputs a random bit b^* as its final output.

Otherwise, at the beginning of the t-th rewinding (in which $\mathcal{S}^*_{\mathsf{trans}}$ expects to generate a signature successfully), the adversary forwards $\mathrm{msg1}_{i,j}$ to the external user instance (holding key $pk_{i,j}$ and message m'_b) in experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$ to receive an answer msg2. The meta-reduction uses this answer in all executions in this row i with first transmission $pk_{i,j},\mathrm{msg1}_{i,j}$. 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$.

We remark that, for each valid execution, $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ (if it has not aborted before) is $V(x',y)\oplus 1$.

Analysis of Adversary $\mathcal{S}^*_{\mathsf{trans}}$. To analyze the success probability of $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathsf{Succ}(\mathcal{H}_k)$ 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):

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

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

First, we collect the cases that $\mathcal{S}^*_{\mathsf{trans}}$ aborts prematurely, returning a random bit b^*. This happens if event $\neg\mathsf{Bound}_{r(n)}$ occurs, if the adversary's guess t for the right repetition has been wrong (event $\neg\mathsf{Guess}$), 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 $\neg\mathsf{SDCh}^+$). 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 $\neg\mathsf{SDCh}^-$).

It is easy to see that the probability for event $\neg\mathsf{SDCh}^+ \vee \neg\mathsf{SDCh}^-$ is negligible by the signature-derivation check; else one can easily build a successful attacker from $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathsf{SDCh}^+ \wedge \mathsf{SDCh}^-$ that all signature-derivation checks return the right answer.

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

$$\begin{aligned}
&\mathrm{Prob}\left[b = b^*\right] \\
&\quad = \mathrm{Prob}\left[b = b^* \;\middle|\; \neg\mathsf{Bound}_{r(n)} \vee \neg\mathsf{Guess}\right] \cdot \mathrm{Prob}\left[\neg\mathsf{Bound}_{r(n)} \vee \neg\mathsf{Guess}\right] \\
&\qquad\quad + \mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}\right] \cdot \mathrm{Prob}\left[\mathsf{Bound}_{r(n)} \wedge \mathsf{Guess}\right] \\
&\quad = \tfrac{1}{2} \cdot \left(1 - \mathrm{Prob}\left[\mathsf{Bound}_{r(n)} \wedge \mathsf{Guess}\right]\right) \\
&\qquad\quad + \mathrm{Prob}\left[\mathsf{Bound}_{r(n)} \wedge \mathsf{Guess}\right] \cdot \mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}\right] \\
&\quad = \tfrac{1}{2} + \mathrm{Prob}\left[\mathsf{Bound}_{r(n)} \wedge \mathsf{Guess}\right] \cdot \left(\mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}\right] - \tfrac{1}{2}\right)
\end{aligned}$$

Next, note that $\mathrm{Prob}\left[\mathsf{Guess} \;\middle|\; \mathsf{Bound}_{r(n)}\right] = 1/r(n)$, 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 $\mathrm{Prob}\left[\mathsf{Bound}_{r(n)}\right] \geq 1 - 1/2p(n)$ we conclude that $\mathrm{Prob}\left[\mathsf{Bound}_{r(n)} \wedge \mathsf{Guess}\right]$ 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:

$$\begin{aligned}
&\mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}\right] - \tfrac{1}{2} \\
&\quad = \mathrm{Prob}\left[b = 1\right] \cdot \mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 1\right] \\
&\qquad\quad + \mathrm{Prob}\left[b = 0\right] \cdot \mathrm{Prob}\left[b = b^* \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 0\right] - \tfrac{1}{2} \\
&\quad = \tfrac{1}{2} \cdot \left(1 - \mathrm{Prob}\left[b^* = 0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 1\right]\right) \\
&\qquad\quad + \tfrac{1}{2} \cdot \mathrm{Prob}\left[b^* = 0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 0\right] - \tfrac{1}{2} \\
&\quad = \tfrac{1}{2} \cdot \left(\mathrm{Prob}\left[b^* = 0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 1\right]\right. \\
&\qquad\quad \left. - \mathrm{Prob}\left[b^* = 0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b = 0\right]\right)
\end{aligned}$$

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

$$\begin{aligned}
&\mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=1\right] - \mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=0\right] \\
&\quad = \sum_{k_0=1}^{q^2} \Big(\mathrm{Prob}\left[b^*=0 \wedge k=k_0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=1\right] \\
&\qquad\qquad - \mathrm{Prob}\left[b^*=0 \wedge k=k_0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=0\right]\Big) \\
&\quad = \tfrac{1}{q^2} \cdot \sum_{k_0=1}^{q^2} \Big(\mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=1, k=k_0\right] \\
&\qquad\qquad - \mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Guess}, \mathsf{Bound}_{r(n)}, b=0, k=k_0\right]\Big)
\end{aligned}$$

The probability that $\mathcal{S}^*_{\mathsf{trans}}$ outputs $b^*=0$, given that the guess is right, the number of repetitions is bounded and $b=1$ (i.e., $\mathcal{S}^*_{\mathsf{trans}}$ forwards the signature generated by Σ) and that $k=k_0$, equals the probability for $\mathsf{Succ}(\mathcal{H}_{k_0})$ (under the condition $\mathsf{Bound}_{r(n)}$). Similarly, under these conditions and that $b=0$, i.e., that $\mathcal{S}^*_{\mathsf{trans}}$ inserts the communication with the user from experiment $\mathsf{trans\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$, the probability for $b^*=0$ is identical to the one for $\mathsf{Succ}(\mathcal{H}_{k_0-1})$ (under the condition $\mathsf{Bound}_{r(n)}$). The latter also relies on Corollary 4.5 (page 10) that Σ succeeds in producing a signature with overwhelming probability if the first execution is valid, because it is guaranteed that $\mathcal{S}^*_{\mathsf{trans}}$ 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 4.5 for simplicity and conclude that

$$\begin{aligned}
&\mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b=1\right] - \mathrm{Prob}\left[b^*=0 \;\middle|\; \mathsf{Bound}_{r(n)}, \mathsf{Guess}, b=0\right] \\
&\quad = \tfrac{1}{q^2} \cdot \sum_{k_0=1}^{q^2} \Big(\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{H}_{k_0}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] - \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{H}_{k_0-1}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\Big) \\
&\quad = \tfrac{1}{q^2} \cdot \Big(\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{H}_{q^2}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] - \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{H}_0) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\Big) \\
&\quad = \tfrac{1}{q^2} \cdot \Big(\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{H}_{q^2})\right] - \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\Big) \\
&\quad = \tfrac{1}{q^2} \cdot \Big(\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{A})\right] - \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\Big)\,.
\end{aligned}$$

where we used the fact that the success probability for the case $\mathcal{H}_{q^2}$ is independent of event $\mathsf{Bound}_{r(n)}$ (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 $\mathrm{Prob}[b=b^*]$, we obtain

$$\begin{aligned}
&\mathrm{Prob}[b=b^*] \\
&\quad \geq \tfrac{1}{2} + \tfrac{1}{2r(n)q^2(n)} \cdot \Big(1 - \tfrac{1}{2p(n)}\Big) \cdot \Big(\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{A})\right] - \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]\Big)\,.
\end{aligned}$$

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 $\mathsf{Bound}_{r(n)}$) it drops to negligible. But then we have derived a successful attacker against the transcript independence of signatures. It follows that our assumption $\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]$ being negligible must have been wrong. Since

$$\begin{aligned}
\mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)})\right] &\geq \mathrm{Prob}\left[\mathsf{Bound}_{r(n)}\right] \cdot \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right] \\
&\geq \Big(1 - \tfrac{1}{2p(n)}\Big) \cdot \mathrm{Prob}\left[\mathsf{Succ}(\mathcal{M}_{r(n)}) \;\middle|\; \mathsf{Bound}_{r(n)}\right]
\end{aligned}$$

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. □

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 $\mathcal{S}^*_{\mathsf{trans}}$, 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 $pk, \mathrm{msg1}$. 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 $\mathcal{S}^*_{\mathsf{trans}}$ could in general not simulate these steps without resetting the external user algorithm.

6 Conclusion

We have shown that for the blind signature schemes of Chaum [Cha83] and of Pointcheval-Stern [PS00] finding security reductions to any non-interactive cryptographic problem in the standard model is hard. This class of cryptographic problems is very broad in the sense that it contains candidates like RSA and collision-resistant hash functions, and also any combination thereof. This also allows us to make stronger infeasibility claims compared to previous results using meta-reductions in other areas.

Concerning optimality of our results we remark that:

- Our result can be transfered to the computational blindness case (under additional stipulations), thus also ruling out many approaches to revert to computationally blindness to circumvent the results for the statistical schemes.
- Enlarging the class of cryptographic problems to interactive ones is too demanding: unforgeability of any blind signature scheme can indeed be securely reduced to an interactive problem in the standard model by simply assuming that the scheme is unforgeable. It is, however, unclear if and how decisional problems can be subsumed under our class of non-interactive (computational) problems.
- Extending the result to protocols with more moves is impossible in light of Okamoto's scheme [Oka06] with four moves in the standard model, based on a non-interactive assumption.

Hence, our result fits well into our current knowledge about constructing blind signatures and shows close boundaries for potential improvements on the efficiency or assumptions.

Acknowledgments

We thank the participants of the Dagstuhl Workshop on Cryptography, in particular Oded Goldreich for bringing up the connection to m-bounded concurrency and Salil Vadhan for pointing out the usage of q-wise independent hash functions instead of pseudorandom functions. We also thank the anonymous reviewers for valuable comments.

References

[Abe01] Masayuki Abe. *A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures*. Advances in Cryptology — Eurocrypt2001, Volume 2045 of Lecture Notes in Computer Science, pages 136–151. Springer-Verlag, 2001.

[AO09] Masayuki Abe and Miyako Ohkubo. *A Framework for Universally Composable Non-Committing Blind Signatures*. Advances in Cryptology — Asiacrypt 2009, Volume 5912 of Lecture Notes in Computer Science, pages 435–450. Springer-Verlag, 2009.

[Bar01] Boaz Barak. *How to Go Beyond the Black-Box Simulation Barrier*. 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*. Proceedings of the Annual Symposium on the Theory of Computing (STOC)2002, pages 484–493. ACM Press, 2002.

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. *Separation Results on the "One-More" Computational Problems*. Topics in Cryptology — Cryptographer's Track, RSA Conference (CT-RSA) 2008, Volume 4964 of Lecture Notes in Computer Science, pages 71–87. Springer-Verlag, 2008.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. *The One-More-RSA-Inversion Problems and the Security of Chaum's Blind Signature Scheme*. *Journal of Cryptology*, 16(3):185–215, 2003.

[Bol03] Alexandra Boldyreva. *Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme*. Public-Key Cryptography (PKC)2003, Volume 2567 of Lecture Notes in Computer Science, pages 31–46. Springer-Verlag, 2003.

[Bro06] Daniel Brown. *What Hashes Make RSA-OAEP Secure?* Number 2006/223 in Cryptology eprint archive. `eprint.iacr.org`, 2006.

[Bro07] Daniel Brown. *Irreducibility to the One-More Evaluation Problems: More May Be Less*. Number 2007/435 in Cryptology eprint archive. `eprint.iacr.org`, 2007.

[BV98] Dan Boneh and Ramarathnam Venkatesan. *Breaking RSA May Be Easier Than Factoring*. Advances in Cryptology — Eurocrypt'98, Lecture Notes in Computer Science, pages 59–71. Springer-Verlag, 1998.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. *Resettable Zero-Knowledge*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2000, pages 235–244. ACM Press, 2000.

[Cha83] David Chaum. *Blind Signatures for Untraceable Payments*. Advances in Cryptology — Crypto'82, pages 199–203. Plemum, New York, 1983.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. *Efficient Blind Signatures Without Random Oracles*. Security in Communication Networks, Volume 3352 of Lecture Notes in Computer Science, pages 134–148. Springer-Verlag, 2004.

[CNS07] Jan Camenisch, Gregory Neven, and Abhi Shelat. *Simulatable Adaptive Oblivious Transfer*. Advances in Cryptology — Eurocrypt'07, Lecture Notes in Computer Science, pages 573–590. Springer-Verlag, 2007.

[Cor02] Jean-Sebastien Coron. *Optimal Security Proofs for PSS and Other Signature Schemes*. Advances in Cryptology — Eurocrypt2002, Volume 2332 of Lecture Notes in Computer Science, pages 272–287. Springer-Verlag, 2002.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. *Concurrent zero-knowledge. J. ACM*, 51(6):851–898, 2004.

[Fis06] Marc Fischlin. *Round-Optimal Composable Blind Signatures in the Common Reference String Model*. Advances in Cryptology - Crypto 2006, Volume 4117 of Lecture Notes in Computer Science, pages 60–77. Springer-Verlag, 2006.

[FS09] Marc Fischlin and Dominique Schröder. *Security of Blind Signatures under Aborts*. Public Key Cryptography, Volume 5443 of Lecture Notes in Computer Science, pages 297–316. Springer-Verlag, 2009.

[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.

[Gol90] Oded Goldreich. *A note on computational indistinguishability. Information Processing Letters*, 34(6):277–281, 1990.

[HILL99] 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.

[HK07] Omer Horvitz and Jonathan Katz. *Universally-Composable Two-Party Computation in Two Rounds*. Advances in Cryptology — Crypto 2007, Lecture Notes in Computer Science, pages 111–129. Springer-Verlag, 2007.

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. *Concurrently-Secure Blind Signatures Without Random Oracles or Setup Assumptions*. Theory of Cryptography Conference (TCC) 2007, Volume 4392 of Lecture Notes in Computer Science, pages 323–341. Springer-Verlag, 2007.

[IR89] Russell Impagliazzo and Steven Rudich. *Limits on the Provable Consequences of One-Way Permutations*. Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1989, pages 44–61. ACM Press, 1989.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. *Security of Blind Digital Signatures*. Advances in Cryptology — Crypto'97, Volume 1294 of Lecture Notes in Computer Science, pages 150–164. Springer-Verlag, 1997.

[KZ06] Aggelos Kiayias and Hong-Sheng Zhou. *Concurrent Blind Signatures Without Random Oracles*. SCN, Volume 4116 of Lecture Notes in Computer Science, pages 49–62. Springer-Verlag, 2006.

[Lin03] Yehuda Lindell. *Bounded-Concurrent Secure Two-Party Computation Without Setup Assumptions*. Proceedings of the Annual Symposium on the Theory of Computing (STOC)2003, pages 683–692. ACM Press, 2003.

[Oka06] Tatsuaki Okamoto. *Efficient Blind and Partially Blind Signatures Without Random Oracles*. Theory of Cryptography Conference (TCC)2006, Volume 3876 of Lecture Notes in Computer Science, pages 80–99. Springer-Verlag, 2006.

[PS00] David Pointcheval and Jacques Stern. *Security Arguments for Digital Signatures and Blind Signatures*. *Journal of Cryptology*, 13(3):361–396, 2000.

[PV06] Pascal Paillier and Jorge Luis Villar. *Trading One-Wayness Against Chosen-Ciphertext Security in Factoring-Based Encryption*. Advances in Cryptology — Asiacrypt'06, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. *Notions of Reducibility between Cryptographic Primitives*. TCC 2004, Lecture Notes in Computer Science, pages 1–20. Springer-Verlag, 2004.

[Sim98] Daniel Simon. *Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?* Advances in Cryptology — Eurocrypt'98, Volume 1403, pages 334–345. Springer-Verlag, 1998.

A Impossibility Result for Computationally Blind Signature Schemes

Here we extend our result to computationally blind signature schemes.

A.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 A.1 (Blind Signature Scheme Relative to an Oracle) *A secure blind signature scheme* $\mathsf{BS} = (\mathsf{KG}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Vf})$ *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* $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*}$ *evaluates to* 1 *is negligibly close to* 1/2, *where*

Experiment $\mathsf{Blind}^{\mathsf{BS}}_{\mathcal{S}^*,\Sigma}(n)$
$(pk, m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{S}^{*,\Sigma}(\mathsf{find}, 1^n)$
$b \leftarrow \{0,1\}$
$\mathsf{st}_{\mathsf{issue}} \leftarrow \mathcal{S}^{*\langle \cdot, \mathcal{U}(pk,m_b)\rangle^1, \langle \cdot, \mathcal{U}(pk,m_{1-b})\rangle^1, \Sigma}(\mathsf{issue}, \mathsf{st}_{\mathsf{find}})$
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 $(\sigma_0, \sigma_1) = (\bot, \bot)$ *if* $\sigma_0 = \bot$ *or* $\sigma_1 = \bot$
$b^* \leftarrow \mathcal{S}^{*,\Sigma}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{st}_{\mathsf{issue}})$
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 A.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* $\mathsf{KeyValCheck}^{\mathsf{BS}}_{\mathcal{S}^*,\mathsf{KVCh}}$ *evaluates to* 1 *is negligible, where*

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

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 A.3 (Generic Forgery Oracle) *For a blind signature scheme* BS *the generic forgery oracle* $\Sigma^c = (\Sigma^c_{\mathsf{sk}}, \Sigma^c_{\mathsf{ex}})$ *consists of two algorithms, where*

Signing Key Generation. *Algorithm* Σ^c_{sk} *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^c_{sk} *returns* $\perp$.

Execution. *Algorithm* Σ^c_{ex} *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^c_{ex} *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 $\Sigma^c_{\mathsf{sk}}(pk, m)$ to derive sk_{Σ^c} and then returns Σ^c_{ex}'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 A.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 4.4) with slight changes. So assume that there exists a signer $\mathcal{S}^*_{\mathsf{trans}}$ in experiment $\mathsf{transc\text{-}ind}^{\mathsf{BS}}_{\mathcal{S}^*_{\mathsf{trans}},\Sigma}(n)$ 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 $\mathcal{S}^*_{\mathsf{blind}}$ against the blindness (with oracle access to Σ^c) as follows. Algorithm $\mathcal{S}^*_{\mathsf{blind}}$ invokes $\mathcal{S}^*_{\mathsf{trans}}(\mathsf{init}, 1^n)$ to get $(pk, \mathsf{st}_1, m_{-1}, m_0)$ and also runs $\Sigma^c_{\mathsf{sk}}(pk, m_0)$ to get sk_{Σ^c}. It outputs $(pk, (\mathsf{st}_1, sk_{\Sigma^c}), m_0, m_0)$ as the initial output in the blindness experiment.

In the following $\mathcal{S}^*_{\mathsf{blind}}$ impersonates the honest user $\mathcal{U}$ for input (pk, m_{-1}) in the left user instance of $\mathcal{S}^*_{\mathsf{trans}}$ by following the user algorithm. In the right user instance for the transcript-independence adversary $\mathcal{S}^*_{\mathsf{blind}}$ 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 $\mathcal{S}^*_{\mathsf{blind}}$ eventually obtains $(m_0, \sigma_0, m_0, \sigma_1)$ (without knowing if σ_0 origins from the communication between $\mathcal{S}^*_{\mathsf{trans}}$ and the user, or from the internally simulated algorithm $\mathcal{S}(sk_{\Sigma^c})$ and the user). If $\sigma_0 = \bot$ then it also sets $\sigma_{-1} \leftarrow \bot$. In any case it forwards $(m_{-1}, \sigma_{-1}, m_0, \sigma_0)$ to $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ 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 $\mathcal{S}^*_{\mathsf{trans}}$ succeeds with non-negligible probability over $1/2$, then $\mathcal{S}^*_{\mathsf{blind}}$ also succeeds with non-negligible probability bounded away from $1/2$. □

Pseudorandom Functions. In order to prove our impossibility result, we take advantage of pseudorandom functions, similar to our deplyoment 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 $\Sigma^{\mathrm{c,aug}}$ which now takes pk, m and randomness s as input and returns σ:

Definition A.5 (Pseudorandom Function Relative to Oracle) *Let $\mathcal{R}_n$ be the set of all functions $f : \{0,1\}^* \to \{0,1\}^{2n+\rho(n)+\psi(n)}$, 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:*

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

where the probability in the first case ist 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 $\mathsf{PRF}(k_1,\cdot),\ldots,\mathsf{PRF}(k_q,\cdot)$ for independent keys $k_1,\ldots,k_q$, or to q independent random functions $f_1,\ldots,f_q$. 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 A.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 executions yields a valid signature, then the magic adversary here also creates another valid signature via $\Sigma^{\text{c,aug}}$. 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 A.6 (Magic Adversary with Access to PRF, Σ^c) *The magic adversary* $\mathcal{A}_{\mathsf{PRF}}$ *for input* pk *and access to the generic forgery oracle* $\Sigma^{c,aug}$ *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*
  $(m_0', m_1', r_0', s_0') \leftarrow \mathsf{PRF}(k, pk||msg1)$ *is generated as the result of the pseudorandom function applied to the public key* pk *and the first message* $msg1$ *of* $\mathcal{S}$; *let* σ_0' *denote the resulting signature.*
if $\mathsf{Vf}(pk, m_0', \sigma_0') = 1$ *then let* $\sigma_1' \leftarrow \Sigma^{c,aug}(pk, m_1'; s_0')$ *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 $\Sigma^{\text{c,aug}}$, pseudorandom functions relative to $\Sigma^{\text{c,aug}}$ exist anyway and therefore do not require an additional assumption. This follows as we then have one-way functions relative to $\Sigma^{\text{c,aug}}$ [Gol90] and can apply the (relativizing) constructions [GGM86, HILL99] to derive pseudorandom functions relative to $\Sigma^{\text{c,aug}}$.

A.2 Impossibility Result

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

Theorem A.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* $\Sigma^{c,aug}$*. 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 A.7 is similar to the proof of Theorem 5.3 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 5.3. 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 $\mathcal{M}$ responses 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 (of Theorem A.7). The proof consists of the following steps. We first show that we can safely replace the pseudorandom function used by $\mathcal{A}$ though 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 $\mathcal{A}_{\mathsf{PRF}}$ be the magic adversary with access to the pseudorandom function PRF and to the generic forgery oracle $\Sigma^{\mathrm{c,aug}}$. We first modify $\mathcal{A}_{\mathsf{PRF}}$ 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 $\mathcal{A}_{\mathsf{PRF}}$ (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 A.5.

The distinguisher D has access to $\Sigma^{\mathrm{c,aug}}$ and to q function oracles $\mathcal{F}_1, \ldots, \mathcal{F}_q$ which accept binary strings as input and return strings of length $2n + \rho(n) + \psi(n)$. The function oracles either compute independent pseudorandom functions $\mathsf{PRF}(k_i, \cdot)$ or truly random functions $f_1, \ldots, f_q$. 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 5.2, 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

$$\mathrm{Prob}\left[\, y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}_f}(y) : V(x', y) = 1 \;\middle|\; \mathcal{A}_f \,\right] \approx 0.$$

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

$$\mathrm{Prob}\left[\, y \leftarrow I(1^n), x' \leftarrow \mathcal{R}^{\mathcal{A}_{\mathsf{PRF}}}(y) : V(x', y) = 1 \;\middle|\; \mathcal{A}_{\mathsf{PRF}} \,\right] \not\approx 0.$$

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

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

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 $(pk, \mathrm{msg1})$ in a row the adversary essentially creates the message-signature pairs independently. For different rows this even holds for the same $pk, \mathrm{msg1}$ as the random function is independent from the ones for the other rows. We can now also switch from $\Sigma^{\mathrm{c,aug}}$ to Σ^{c} as long as we guarantee that Σ^{c} gives identical answers for executions with the same $pk, \mathrm{msg1}$; 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 5.3 go thorugh in this case.

The meta-reduction $\mathcal{M}$ here essentially behaves as the one in the proof of Theorem 5.3, 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 5.3. 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 4.5 to ensure that a failing Σ^{c} in the transcript-independence experiment does not prevent the adversary $\mathcal{S}^{*}_{\mathsf{trans}}$ 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. □