An extended abstract of this paper appears at Eurocrypt 2013.

Limitations of the Meta-Reduction Technique: The Case of Schnorr Signatures

Marc Fischlin[1] Nils Fleischhacker[2]

[1]Technische Universität Darmstadt
www.cryptoplexity.de
[2]Saarland University
ca.cs.uni-saarland.de

Abstract. We revisit the security of Fiat-Shamir signatures in the non-programmable random oracle model. The well-known proof by Pointcheval and Stern for such signature schemes (Journal of Cryptology, 2000) relies on the ability to re-program the random oracle, and it has been unknown if this property is inherent. Pailler and Vergnaud (Asiacrypt 2005) gave some first evidence of the hardness by showing via meta-reduction techniques that algebraic reductions cannot succeed in reducing key-only attacks against unforgeability to the discrete-log assumptions. We also use meta-reductions to show that the security of Schnorr signatures cannot be proven equivalent to the discrete logarithm problem without programming the random oracle. Our result also holds under the one-more discrete logarithm assumption but applies to a large class of reductions, we call *single-instance* reductions, subsuming those used in previous proofs of security in the (programmable) random oracle model. In contrast to algebraic reductions, our class allows arbitrary operations, but can only invoke a single resettable adversary instance, making our class incomparable to algebraic reductions.

Our main result, however, is about meta-reductions and the question if this technique can be used to further strengthen the separations above. Our answer is negative. We present, to the best of our knowledge for the first time, limitations of the meta-reduction technique in the sense that finding a meta-reduction for general reductions is most likely infeasible. In fact, we prove that finding a meta-reduction against a potential reduction is equivalent to finding a "meta-meta-reduction" against the strong existential unforgeability of the signature scheme. This means that the existence of a meta-reduction implies that the scheme must be insecure (against a slightly stronger attack) in the first place.

1 Introduction

On a technical level, we investigate the security of Fiat-Shamir (FS) signatures [FS87] in the non-programmable random oracle model (NPROM), i.e., where programming the hash function is prohibited. Such programming has been exploited in the security proof for common FS signatures by Pointcheval and Stern [PS00], bringing forward the question if the security result remains valid in the more stringent model of non-programmable random oracles. Conceptually, though, the more interesting result in the paper refers to limitations of so-called meta-reductions. Such meta-reductions are also called "reductions against the reductions" as they basically treat the reduction as an adversary itself and reduce the existence of such a reduction to a presumably hard problem, ruling out reductions and therefore security proofs for the underlying scheme. This proof technique recently gained quite some attention as it rules out certain reductions, especially those which only treat the adversary but not the underlying primitive as a black box (e.g., [Cor02, PV05, GBL08, FS10, Pas11, GW11, AGO11, Seu12]). We show, via a "meta-meta-reduction", that one cannot use the meta-reduction technique to show impossibility results for FS signatures in the NPROM.

1.1 Fiat-Shamir Signatures in the NPROM

The class of FS signatures comprises all transformed three-move identification schemes in which the challenge $\textsc{ch}$, sent by the verifier in return of the prover's initial commitment $\textsc{com}$, is replaced by the hash value $H(\textsc{com}, m)$ for message m. The prover's response $\textsc{resp}$, together with $\textsc{com}$, then yields the signature $(\textsc{com}, \textsc{resp})$ for m. For some cases, like the Schnorr signature scheme [Sch91], the signature can be shortened by using $(\textsc{ch}, \textsc{resp})$ instead.

The common security proof for FS signature schemes in the random oracle model [BR93], given in [PS00], basically works as follows. The reduction to the underlying problem, such as the discrete logarithm problem for the Schnorr scheme, runs the adversary twice. In the first runs the reduction gets a signature forgery $(\textsc{com}, \textsc{resp})$ for message m and challenge $\textsc{ch} = H(\textsc{com}, m)$. In the second run it re-programs H to yield a distinct challenge $\textsc{ch}' = H(\textsc{com}, m)$ and response $\textsc{resp}'$. From both signatures the reduction can then compute a solution to the underlying problem. Clearly, this technique relies on the programmability of the hash function.[1]

Fischlin et al. [FLR$^+$10] later defined reductions in the non-programmable random oracle model (NPROM) by externalizing the hash function to both the adversary *and* the reduction.[2] In the NPROM the reduction may still observe the adversary's queries to the hash function, but cannot change the reply. Obviously, this non-programming property matches much closer our understanding of "real" hash functions and instantiations through, say, SHA-3. Interestingly, though, Fischlin et al. [FLR$^+$10] do not investigate this arguably most prominent application of the random oracle methodology. Instead, they separate programming and non-programming reductions (and an intermediate notion called weakly programming reductions) through the case of OAEP encryption, FDH signatures, and trapdoor permutation based KEMs. Weakly programming reductions are allowed to reset the random oracle and redirect the value to some (external) random answer. Our first result is to formally confirm the intuition that FS signatures should still be secure in the weakly programmable random oracle model.

1.2 Limitations Through Meta-Reductions

The more interesting question is if FS signatures can be shown to be secure in the NPROM. Our first result in this regard is negative and applies to discrete log schemes like the Schnorr signature scheme [Sch91] or the RSA-based Guillou-Quisquater scheme [GQ88]. Namely, we first consider any reduction $\mathcal{R}$ which initiates only a single (black-box) instance of the adversary $\mathcal{A}$ for some public key pk, but such that it can reset $\mathcal{A}$ arbitrarily to the point after having handed over the public key (from the fixed group). Note that the reduction in the *programmable* ROM in [PS00] is of this kind, only that it can also change the behavior of the random oracle, unlike our reductions here in the NPROM. We show that this type of *single-instance* reduction to the discrete log problem cannot succeed in the NPROM under the one-more discrete log assumption [BNPS03].

Our impossibility result follows from presenting a meta-reduction $\mathcal{M}$ against $\mathcal{R}$. That is, we show that if one can find a reduction $\mathcal{R}$ which successfully solves the DL problem given black-box access to any successful adversary $\mathcal{A}$ against the signature scheme, then there is a meta-reduction $\mathcal{M}$ breaking the one-more DL problem directly. Since we also present a successful (unbounded) adversary $\mathcal{A}$ which $\mathcal{M}$ can simulate towards $\mathcal{R}$ efficiently, we conclude that the existence of reduction $\mathcal{R}$ would already contradict the one-more DL problem.

We observe that our meta-reduction, too, works in the NPROM and thus cannot program the random oracle for $\mathcal{R}$; else the meta-reduction would violate the idea of modeling hash functions as non-programmable. It is also easy to show that, if the meta-reduction, unlike the reduction, was allowed to program the random oracle, this "unfair" situation would straightforwardly dismiss the possibility of such reductions. However,

[1]Note that this proof reduces the security of the signature scheme to the underlying number-theoretic problem via special soundness. Abdalla et al. [AABN02] more generally consider FS schemes with reductions to the identification schemes. We do not cover the latter type of reductions and schemes here.

[2]The role of programmability was first investigated by Nielsen [Nie02], even though not for reductions as in the proofs of Fiat-Shamir schemes.

such an approach seems to violate the idea behind non-programmable oracles as a mean to capture real-world hash functions over which no party, not even the meta-reduction, has control.

The noteworthy property of our meta-reduction $\mathcal{M}$ is that, unlike most of the previous proposals (cf. [Fis12]), it does not work by resetting the reduction $\mathcal{R}$. The reset strategy is usually used to rewind the reduction and, in case of signature schemes, get an additional signature through a signing query in an execution branch, and display this signature back to $\mathcal{R}$ as a forgery in the main branch. However, this means that one needs to take care of correlations between the additional signature and the reduction's state. Instead of using such resets, our meta-reduction will essentially run two independent copies of the reduction and use the signatures of one execution in the other one. The independence of the executions thus "decorrelates" the additional signature from the reduction's state, avoiding many complications from the resetting strategy.

1.3 Limitations of Meta-Reductions

Does our meta-reduction impossibility result for non-programming reductions extend to other cases like the discrete logarithm problem? We show that this is unlikely, thus showing limitations of the meta-reduction technique. The idea is to consider the meta-reduction itself as a reduction, and to use the meta-reduction technique against this reduction. Hence, we obtain a "meta-meta-reduction" $\mathcal{N}$ which now simulates the reduction $\mathcal{R}$ for $\mathcal{M}$, just as the meta-reduction simulates the adversary for $\mathcal{R}$.

More concretely, assume that we consider reductions $\mathcal{R}$ transforming an adversary $\mathcal{A}$ against the signature scheme in a black-box way into a solver for some cryptographic problem $\Pi_{\mathcal{R}}$. Then, a meta-reduction $\mathcal{M}$ should turn $\mathcal{R}$ (to which it has black-box access) into a successful solver for some problem $\Pi_{\mathcal{M}}$. For technical reasons, in our case this problem $\Pi_{\mathcal{M}}$ has to be non-interactive, e.g., correspond to the discrete logarithm problem; this also circumvents the case of our previous meta-reduction for the interactive one-more DL problem. Then we show that such a meta-reduction can be used to build a meta-meta-reduction $\mathcal{N}$ against the strong unforgeabilty of the signature scheme.

In other words, the meta-reduction technique cannot help to rule out black-box reductions to arbitrary problems, unless the signature is insecure in the first place. Here, insecurity refers to the notion of strong unforgeability where the adversary also succeeds by outputting a new signature to a previously signed message. In fact, in the programmable ROM the security proof in [PS00] actually shows that the FS schemes achieve this stronger notion.

1.4 Related Work

As mentioned before, meta-reductions have been used in several recent results to rule out black-box reductions for Fiat-Shamir schemes, and especially for Schnorr signatures. Paillier and Vergnaud [PV05] analyzed the security of Schnorr Signatures in the standard model. They showed with the help of meta-reductions that, if the one-more discrete logarithm assumption holds, the security of Schnorr signatures cannot be reduced to the (one-more) discrete logarithm problem, at least using algebraic reductions. While algebraic reductions where first defined by Boneh and Venkatesan [BV98], Paillier and Vergnaud [PV05], however, use a slightly more liberal definition of algebraicity. Their notion basically states that, given the discrete logarithm of all of the reduction's inputs and access to the reduction, it is possible to compute the discrete logarithm of any group element output by the reduction. We note that the ability to trace the discrete logarithms of the group elements produced by the reduction is important to their result and allows them to prove impossibility even for key-only attacks.

Paillier and Vergnaud [PV05] also extended their result to other signature schemes, including the Guillou-Quisquater scheme [GQ88] and the one-more RSA assumption [BNPS03]. They also considered the tightness loss in the Pointcheval-Stern proof for the Schnorr signature scheme in the programmable random oracle model. They showed, again for algebraic reductions, that the security loss of a factor $\sqrt{q_H}$ is inevitable, where q_H is the maximum number of random oracle queries by the adversary. This bound was later raised to $q_H^{2/3}$ by Garg et al. [GBL08] in the same setting. Seurin [Seu12] recently improved this bound further to $\mathcal{O}(q_H)$. Using meta-reduction techniques and considering algebraic reductions, too, he proved it is unlikely that a tighter reduction exists.

In a recent work, Baldimtsi and Lysyanskaya [BL12] showed, via meta-reductions, that one cannot prove blind Schnorr signatures secure via black-box reductions. Their meta-reduction, like ours here, has the interesting feature of being non-resetting. Remarkably, though, they seem to rule out the more liberal *programming* reductions, whereas our result is against the "more confined" *non-programming* reductions. However, their result considers a special type of programming reduction, called naive. This roughly means that one can predict the reduction's programmed random oracle answers by reading the reduction's random tape. This property is inherently tied to the programmability and is clearly not fulfilled by non-programmable, external random oracles; for such oracles even the reduction does not know the answers in advance. This, unfortunately, also means that their meta-reduction technique may not apply to non-programmable hash functions. In other words, one may be able to bypass their impossibility result and may still be able to find a cryptographic security proof for such schemes, by switching to the non-programmable random oracle model, or even to standard-model hash functions.

1.5 Organization

In Section 2 we first recall some basic facts about signatures and (general and discrete-log specific) cryptographic problems. Then we show that FS signatures are secure in the weakly programmable random oracle model, and prove our meta-reduction impossiblity result for single-instance reductions in the NPROM in Section 3. Our main result about meta-meta-reductions appears in Section 4.

2 Preliminaries

We use standard notions for digital signature schemes $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ such as existential unforgeability and strong existential unforgeability. We usually assume (non-trivially) randomized signature schemes, where the signature algorithm has super-logarithmic min-entropy for the security parameter κ, i.e., $H_\infty(\mathsf{Sign}(\mathsf{sk}, m)) \in \omega(\log(\kappa))$ for all keys sk, all messages m, and given the random oracle.

2.1 Digital Signature Schemes

We recall the syntax of digital signature schemes in Definition 2.1.

Definition 2.1 (Digital Signature Scheme) *A signature scheme* $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of three algorithms:*

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

The signing algorithm Sign *takes as input a secret key* sk *and a message* $m \in \{0,1\}^*$. *Additionally it takes a randomizer* $\omega \in \mathcal{C}oins_{\mathsf{pk}}$ *and outputs a signature* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; \omega)$. *The set* $\mathcal{C}oins_{\mathsf{pk}}$ *depends on the signature scheme and possibly on the public key. Whenever* ω *is not specified in an invocation of* Sign*, it is to be understood as uniformly chosen from* $\mathcal{C}oins_{\mathsf{pk}}$.

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

The scheme $\mathcal{S}$ *is* correct *if and only if for all* $\kappa \in \mathbb{N}$*, all* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$*, all* $m \in \{0,1\}^*$*, all* $\omega \in \mathcal{C}oins_{\mathsf{pk}}$*, and* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; \omega)$*, it holds that* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) \overset{?}{=} 1$.

We recall two common security notions for digital signature schemes. Namely the notions of existential unforgeability under adaptive chosen-message attacks in Definition 2.2 and strong existential unforgeability under adaptive chosen-message attacks in Definition 2.3.

Definition 2.2 (Existential Unforgeability) *A signature scheme* $\mathcal{S}$ *is said to be* existentially unforgeable under adaptive chosen-message attacks ($\mathsf{EUF\text{-}CMA}$) *if and only if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *the following success function is negligible in* κ.

$$\mathsf{Succ}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa) = \Pr\left[\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa) \overset{?}{=} 1\right]$$

where $\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa)$ *is the* existential unforgeability experiment *from Figure 1.*

Definition 2.3 (Strong Existential Unforgeability) *A signature scheme* $\mathcal{S}$ *is said to be* strongly existentially unforgeable under adaptive chosen-message attacks (sEUF-CMA) *if and only if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *the following success function is negligible in* κ*.*

$$\mathsf{Succ}^{\mathcal{S},\mathcal{A}}_{\mathsf{sEUF\text{-}CMA}}(\kappa) = \Pr\left[\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{sEUF\text{-}CMA}}(\kappa) \stackrel{?}{=} 1\right]$$

where $\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{sEUF\text{-}CMA}}(\kappa)$ *is the* strong existential unforgeability experiment *from Figure 1.*

```
$\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa)$ :
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{S}.\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
  if $\begin{pmatrix} \mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \stackrel{?}{=} 1 \\ \text{and } m^* \notin \mathbb{M} \end{pmatrix}$
    then output 1
    else output 0

$\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{sEUF\text{-}CMA}}(\kappa)$ :
  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$
  $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{S}.\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
  if $\begin{pmatrix} \mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \stackrel{?}{=} 1 \\ \text{and } (m^*, \sigma^*) \notin \mathbb{Q} \end{pmatrix}$
    then output 1
    else output 0
```

Figure 1: (Strong) Existential Unforgeability Experiments. We denote by $\mathbb{M}$ the set of all messages queried by $\mathcal{A}$ to the signing oracle and by $\mathbb{Q}$ the set of message-signature pairs resulting from sign queries issued by $\mathcal{A}$.

For the most part, we focus on non-trivially randomized signature schemes in this paper. We introduce the notion of *randomized* signature schemes in Definition 2.5. As the definition is in terms of the signatures' min-entropy, we first recall the definition of a discrete random variable's min-entropy in Definition 2.4.

Definition 2.4 (Min-Entropy) *The min-entropy of a discrete random variable* X *is defined as*

$$H_\infty(X) = -\log \max_i p_i$$

where p_i *for* $1 \leq i \leq N$ *is the probability that* x_i *is chosen according to the distribution of* X*. The min-entropy gives us an upper bound to the probability that a value chosen according to the distribution of* X *matches any fixed value as*

$$\forall x : \Pr[x' = x | x' \leftarrow X] \leq 2^{-H_\infty(X)}.$$

Definition 2.5 (Randomized Signature Scheme) *A signature scheme* $\mathcal{S}$ *is said to be* randomized *if and only if for all security parameters* $\kappa \in \mathbb{N}$*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$*, and all messages* $m \in \{0,1\}^*$ *the min-entropy of the random variable describing the signing process is super-logarithmic in the security parameter. That is, the following must hold:*

$$\forall \kappa \in \mathbb{N} : \forall (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa) : \forall m \in \{0,1\}^* : H_\infty(\mathsf{Sign}(\mathsf{sk}, m)) \in \omega(\log(\kappa)).$$

For random oracle based schemes we also assume that the entropy is large when given the random oracle. Schnorr signatures obey this property of randomized signatures, but they also allow to determine the hash queries which the signer has made given the signature only:

Definition 2.6 (Signature Scheme with Reconstructible Hash Queries) *A signature scheme* $\mathcal{S}$ *in the random oracle model is said to have* reconstructible hash queries *if there exists an efficient algorithm* $\mathcal{A}$ *such that for all security parameters* $\kappa \in \mathbb{N}$*, all key pairs* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^\kappa)$*, all messages* $m \in \{0,1\}^*$*, and all signatures* $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *for which* Sign *made an (ordered) sequence* Q *of random oracle queries, we have* $Q = \mathcal{A}(\mathsf{pk}, m, \sigma)$*.*

2.2 Cryptographic Problems

We define a cryptographic problem as a game between a challenger and an adversary. The challenger uses an instance generator to generate a fresh instance of the problem. The adversary is then supposed to find a solution for said instance. The challenger may assist the adversary by providing access to some oracle, like a decryption oracle in a chosen-ciphertext attack against indistinguishability. Eventually the adversary outputs a solution for the problem instance and the challenger uses a verification algorithm to check whether the solution is correct.

For many problems there exist trivial adversaries, e.g., succeeding in an indistinguishability game by pure guessing. One is usually interested in the advantage of adversaries beyond such trivial strategies. We therefore introduce a so-called *threshold algorithm* to cover such trivial attacks and measure any adversary against this threshold adversary.

Definition 2.7 (Cryptographic Problem) *A cryptographic problem* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *consists of four algorithms:*

- *The instance generator* IGen *takes as input the security parameter* 1^κ *and outputs a problem instance* z*. The set of all possible instances output by* IGen *is called* $\mathbf{Inst}$*.*
- *The computationally unbounded and stateful oracle algorithm* Orcl *takes as input a query* $q \in \{0,1\}^*$ *and outputs a response* $r \in \{0,1\}^*$ *or a special symbol* $\perp$ *indicating that* q *was not a valid query.*
- *The deterministic verification algorithm* Vrfy *takes as input a problem instance* $z \in \mathbf{Inst}$ *and a candidate solution* $x \in \mathbf{Sol}$*. The algorithm outputs* $b \in \{0,1\}$*. We say* x *is a valid solution to instance* z *if and only if* $b \overset{?}{=} 1$*.*
- *The efficient threshold algorithm* Thresh *takes as input a problem instance* z *and outputs some* x*. The threshold algorithm is a special adversary and as such also has access to* Orcl*.*

We note that the algorithms IGen*,* Orcl*,* Vrfy *potentially have access to shared state that persists for the duration of an experiment.*

Definition 2.8 (Hard Cryptographic Problem) *For a cryptographic problem* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *and an adversary* $\mathcal{A}$ *we define the following experiment:*

$$\mathsf{Exp}_\Pi^{\mathcal{A}}(\kappa) : [z \leftarrow \mathsf{IGen}(1^\kappa); x \leftarrow \mathcal{A}^{\mathsf{Orcl}}(z); b \leftarrow \mathsf{Vrfy}(z, x); \text{output } b].$$

The problem Π *is said to be* hard *if and only if for all probabilistic polynomial-time algorithms* $\mathcal{A}$ *the following advantage function is negligible in the security parameter* κ*:*

$$\mathsf{Adv}_\Pi^{\mathcal{A}}(\kappa) = \Pr\left[\mathsf{Exp}_\Pi^{\mathcal{A}}(\kappa) \overset{?}{=} 1\right] - \Pr\left[\mathsf{Exp}_\Pi^{\mathsf{Thresh}}(\kappa) \overset{?}{=} 1\right],$$

where the probability is taken over the random tapes of IGen *and* $\mathcal{A}$*.*

We sometimes require some additional properties of cryptographic problems, summarized in the following definition:

Definition 2.9 (Specific Cryptographic Problems) *Let* $\Pi = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy}, \mathsf{Thresh})$ *be a cryptographic problem as defined in Definition 2.7.*

- *The problem* Π *is said to be* non-interactive *if and only if* $\Pi.\mathsf{Orcl}$ *is the algorithm that always outputs* $\perp$ *and never changes the shared state.*
- *The problem* Π *is said to be* efficiently generatable *if and only if* $\Pi.\mathsf{IGen}$ *is a polynomial-time algorithm.*

- *The problem* Π *is said to be* solvable *if and only if* $\Pi.\mathbf{Sol}$ *is recursively enumerable, and the following holds:*
$$\forall z \leftarrow \Pi.\mathsf{IGen}(1^\kappa) : (\exists x \in \Pi.\mathbf{Sol} : \Pi.\mathsf{Vrfy}(z,x) \stackrel{?}{=} 1).$$
- *The problem* Π *is said to be* monotone *if and only if for all instances* $z \leftarrow \Pi.\mathsf{IGen}(1^\kappa)$*, all solutions* $x \in \Pi.\mathbf{Sol}$*, all* $n \in \mathbb{N}$*, and all sequences of queries* $(q_1, \ldots, q_n)$ *the following holds: If* $\Pi.\mathsf{Vrfy}(z,x) \stackrel{?}{=} 1$ *holds after executing the queries* $\Pi.\mathsf{Orcl}(q_1); \ldots ; \Pi.\mathsf{Orcl}(q_n)$*, then this already held before* $\Pi.\mathsf{Orcl}(q_n)$ *was executed.*

Intuitively, an algorithm solving a monotone problem is not punished for issuing fewer queries. In particular, if a solution is valid after some sequence of queries, it is also valid if no queries were executed at all.

2.3 Discrete Logarithm Assumptions

The discrete logarithm problem with its corresponding hardness assumption is a specific instance of a non-interactive, efficiently generatable, and solvable problem. The assumption about the computational infeasibility of computing logarithms in certain groups is formally defined in Definition 2.10.

Definition 2.10 (Discrete Logarithm Assumption) *Let* $\mathbb{G} = \langle g \rangle$ *be a group of prime order* q *with* $|q| = \kappa$*. The* discrete logarithm (DL) *problem over* $\mathbb{G}$ *—written* $\mathsf{DL}_\mathbb{G}$*— is defined as follows:*

Instance and Solution space: *The instance space* **Inst** *is* $\mathbb{G}$ *and the solution space* **Sol** *is* $\mathbb{Z}_q$*.*

Instance Generation: *The instance generator* $\mathsf{IGen}(1^\kappa)$ *chooses* $z \xleftarrow{\$} \mathbb{G}$ *and outputs* z*. Note that this sampling of* z *may require to pick a random* $w \xleftarrow{\$} \mathbb{Z}_q$ *and compute* $z = g^w$*.*

Verification: *The verification algorithm* $\mathsf{Vrfy}(z,x)$ *computes* $z' = g^x$*. If* $z' \stackrel{?}{=} z$*, then it outputs* 1*, otherwise it outputs* 0*.*

Threshold: *The threshold algorithm* $\mathsf{Thresh}(z)$ *chooses* $x \xleftarrow{\$} \mathbb{Z}_q$ *and outputs* x*.*

The discrete logarithm *assumption is said to hold over* $\mathbb{G}$ *if* $\mathsf{DL}_\mathbb{G}$ *is hard.*

A natural extension of the discrete logarithm problem are the interactive, efficiently generatable, monotone, and solvable *one-more discrete logarithm* problems first introduced by Bellare et al. [BNPS03]. They are interactive, as the adversary is given access to an oracle capable of solving the $\mathsf{DL}_\mathbb{G}$ problem. However, an adversary computing $n+1$ discrete logarithms can only request at most n discrete logarithms from the DL oracle, hence, the name *one-more* discrete-log problem. The problems with their corresponding hardness assumptions are formally described in Definition 2.11. The assumptions are believed to be stronger than the regular DL assumption [BMV08].

Definition 2.11 (n-One-More Discrete Logarithm Assumption [BNPS03]) *Let* $\mathbb{G} = \langle g \rangle$ *be a group of prime order* q *with* $|q| = \kappa$*. The* n-one-more discrete logarithm (n-DL) *problem over* $\mathbb{G}$ *–written* n-$\mathsf{DL}_\mathbb{G}$*– is defined as follows:*

Instance and Solution space: *The instance space* **Inst** *is* $\mathbb{G}^{n+1}$ *and the solution space* **Sol** *is* $\mathbb{Z}_q^{n+1}$*.*

Shared State: *The shared state consists only of a single counter variable* i*.*

Instance Generation: *The instance generator* $\mathsf{IGen}(1^\kappa)$ *initializes* $i := 0$ *in the shared state, chooses* $z_0, \ldots, z_n \xleftarrow{\$} \mathbb{G}$*, and outputs* $(z_0, \ldots, z_n)$*.*

Oracles: *The oracle algorithm* $\mathsf{Orcl}(z)$*, on input* $z \in \mathbb{G}$*, increments* $i := i + 1$*. It then exhaustively searches* $\mathbb{Z}_q$ *for an* x *such that* $g^x \stackrel{?}{=} z$ *and outputs* x*. On input some* $z \notin \mathbb{G}$*,* Orcl *outputs* $\perp$*.*

Verification: *The verification algorithm* $\mathsf{Vrfy}((z_0,\ldots,z_n),(x_0,\ldots,x_n))$ *computes* $z'_j = g^{x_j}$. *If* $z'_j \stackrel{?}{=} z_j$ *for all* j *and if* $i \leq n$, *then it outputs* 1, *otherwise it outputs* 0.

Threshold: *The threshold algorithm* $\mathsf{Thresh}(z)$ *chooses* $x_0,\ldots,x_n \xleftarrow{\$} \mathbb{Z}_q$ *and outputs* $(x_0,\ldots,x_n)$.

The n-one-more discrete logarithm (n-DL) *assumption is said to hold over* $\mathbb{G}$, *if and only if the* n-$\mathsf{DL}_{\mathbb{G}}$ *problem is hard.*

3 Security of Schnorr Signatures

We first recall the definition of the Schnorr signature scheme (SSS) [Sch90, Sch91] as derived from the Schnorr identification scheme via the Fiat-Shamir transform [FS87]. Afterwards, we analyze the security of the resulting signature scheme in two variants of the random oracle model, in which reductions are limited in the way they can program the random oracle.

Definition 3.1 (Schnorr Signature Scheme) *Let* $\mathbb{G}$ *be a cyclic group of prime order* q *with generator* g *and let* $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_q$ *be a hash function modeled as a random oracle. The Schnorr signature scheme, working over* $\mathbb{G}$, *is defined as follows:*

Key Generation: *The key generation algorithm* $\mathsf{KGen}(1^\kappa)$ *proceeds as follows: Pick* $\mathsf{sk} \xleftarrow{\$} \mathbb{Z}_q$, *compute* $\mathsf{pk} := g^{\mathsf{sk}}$, *and output* $(\mathsf{sk},\mathsf{pk})$.

Signature Generation: *The signing algorithm* $\mathsf{Sign}(\mathsf{sk}, m; r)$ *proceeds as follows: Use* $r \in \mathbb{Z}_q$ *and compute* $R := g^r$. *Compute* $c := \mathcal{H}(R, m)$ *and* $y := r + \mathsf{sk} \cdot c \mod q$. *Output* $\sigma := (c, y)$.

Signature Verification *The verification algorithm* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$ *proceeds as follows: Parse* σ *as* (c, y). *If* $c \stackrel{?}{=} \mathcal{H}(\mathsf{pk}^{-c} g^y, m)$, *then output* 1, *otherwise output* 0.

3.1 Unforgeability of Schnorr Signatures Under Randomly Programming Reductions

We begin by showing that the original proof by Pointcheval and Stern [PS96, PS00] still holds for randomly programming reductions. Randomly programming reductions as defined in [FLR+10] do not simulate the random oracle themselves. Instead, they can re-set the random oracle to another hash value. As shown in [FLR+10] such randomly programming reductions are equivalent to the weakly-programmable random oracle model (WPROM) which is in between the programmable and non-programmable ROM. Whereas a conventional random oracle has only a single interface implementing a random mapping from domain **Dom** to range **Rng**, a weakly programmable random oracle has three interfaces, which allow for programming but only in a weak sense: one cannot freely re-program the hash values but only re-set them to another random value:

Definition 3.2 (Weakly Programmable Random Oracle) *A* weakly programmable random oracle *(WPRO) exposes three interfaces to the caller:*

Evaluation: *The evaluation interface* RO_{eval} *behaves as a conventional random oracle, mapping* $\mathbf{Dom} \to \mathbf{Rng}$.

Random: *The random interface* RO_{rand} *takes as input bit strings of arbitrary length and implements a random mapping* $\{0,1\}^* \to \mathbf{Rng}$.

Programming: *The programming interface* RO_{prog} *takes as input a pair* $(a, b) \in \mathbf{Dom} \times \{0,1\}^*$ *and programs* $\mathsf{RO}_{eval}(a)$ *to evaluate to* $\mathsf{RO}_{rand}(b)$.

The randomly programming reduction gets oracle access to all three interfaces, whereas the adversary only gets access to the RO_{eval} interface. We now show that randomly programming reductions are sufficient to prove SSS secure in the ROM.

Theorem 3.3 ($\mathsf{EUF\text{-}CMA}$ Security of SSS Under Randomly Programming Reductions) *The* $\mathsf{EUF\text{-}CMA}$ *security of* SSS *is reducible to the* discrete logarithm *problem over* $\mathbb{G}$ *using a randomly programming reduction* $\mathcal{R}$.

Proof. (Sketch) We provide a description of a randomly programming reduction. Basically, the reduction behaves identically to the original reduction from [PS00]. We merely need to show that the limited programmability still allows the reduction to simulate the signing oracle and to apply the forking technique.

Simulating the Signing Oracle. When asked to sign a message m, the reduction $\mathcal{R}$ chooses $b \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and $y \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $c \leftarrow \mathsf{RO}_{rand}(b)$ and $R := \mathsf{pk}^{-c}g^y$, and reprograms the oracle by calling $\mathsf{RO}_{prog}((R,m),b)$. It then outputs (c,y).

Forking. To fork after the first j oracle calls, $\mathcal{R}$ proceeds as follows: For $i \geq j$, whenever $\mathcal{A}$ asks the query $\mathsf{RO}_{eval}(a_i)$, $\mathcal{R}$ picks $b_i \overset{\$}{\leftarrow} \{0,1\}^\kappa$ and reprograms the oracle by calling $\mathsf{RO}_{prog}(a_i,b_i)$ before the original query is evaluated. Note that, just as in case of the full programming of the random oracle in [PS00] for the random values R from the group, the distribution of $\mathsf{RO}_{eval}(a_i)$ is slightly biased, because of the possibility that the b_i's collide. However, as the size of the domain from which b_i's are uniformly chosen is exponential in the security parameter, the collision probability —and therefore the bias— is also negligible, as it is for random group elements. $\square$

We thus show that the limited programmability of a randomly programming random oracle is sufficient to obtain a (loose) proof of security for Schnorr signatures. In particular, choosing range points of the random oracle at will is not required for the proof. We note that the above result transfers to other FS schemes such as [Oka93, GQ88, FF02].

3.2 Schnorr Signatures are not Provably Secure Under Non-Programming Single-Instance Reductions

We now show that the Schnorr Signature Scheme cannot be proven existentially unforgeable under chosen message attacks without programming the random oracle —at least with respect to a slightly restricted type of reduction. We actually prove that, if such a reduction exists, the 1-one-more discrete logarithm assumption does not hold over $\mathbb{G}$.

We term the restricted class of reductions as *single-instance reductions*. Such single-instance reductions only invoke a single instance of the adversary and, while they may rewind the adversary, they may not rewind it to a point before it received the public key for the first time. This class of reductions is especially relevant, because both the original security reduction by Pointcheval and Stern [PS00] as well as the one in Theorem 3.3 are of this type.

Instead of simulating the random oracle itself, a non-programming reduction works relative to an external fixed random function and it is required to honestly answer all random oracle queries. That is, the black-box reduction can observe the adversary's queries to the random oracle, but cannot change the answers. We omit a formal approach (see [FLR$^+$10]) because the definition reflects the intuition straightforwardly. We remark that the approach assumes fully-black-box reductions [RTV04] (or, in terms of the CAP taxonomy of [BBF13], the BBB-type of reduction) which need to work for any (unbounded) adversary oracle. In particular, and we will in fact exploit this below, the adversary can thus depend on the reduction. We may therefore think of the adversary as a family $\mathbb{A}$ of adversaries $\mathcal{A}_{\mathcal{R},a}$, depending on the reduction $\mathcal{R}$ and using some randomness a. We believe it is conceptually easier in this case here to make the randomness a explicit, as opposed to having a single adversary that internally chooses a at the beginning of the execution. It is nonetheless sometimes convenient to omit these subindices and to simply write $\mathcal{A}$.

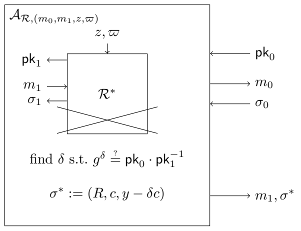

$\mathcal{A}_{\mathcal{R},(m_0,m_1,z,\varpi)}$
z, ϖ
pk_1
m_1
σ_1
$\mathcal{R}^*$
pk_0
m_0
σ_0
find δ s.t. $g^\delta \stackrel{?}{=} \mathsf{pk}_0 \cdot \mathsf{pk}_1^{-1}$
$\sigma^* := (R, c, y - \delta c)$
m_1, σ^*

Figure 2: For each reduction $\mathcal{R}$, the associated inefficient adversary $\mathcal{A}_{\mathcal{R}}$ works by choosing messages, a group element, and a random tape describing a random function. It proceeds to forge a signature by internally simulating an instance of $\mathcal{R}$ on the group element and the random tape and using its unbounded computational power to adapt the resulting signature to the public key passed by the outside game.

Theorem 3.4 (Non-Programming Irreducibility for SSS) *Assume that the* 1-one-more discrete logarithm *assumption holds over $\mathbb{G}$. Then, there exists no non-programming single-instance fully-black-box reduction that reduces the* $\mathsf{EUF\text{-}CMA}$ *security of* SSS *over $\mathbb{G}$ to the discrete logarithm problem over $\mathbb{G}$.*

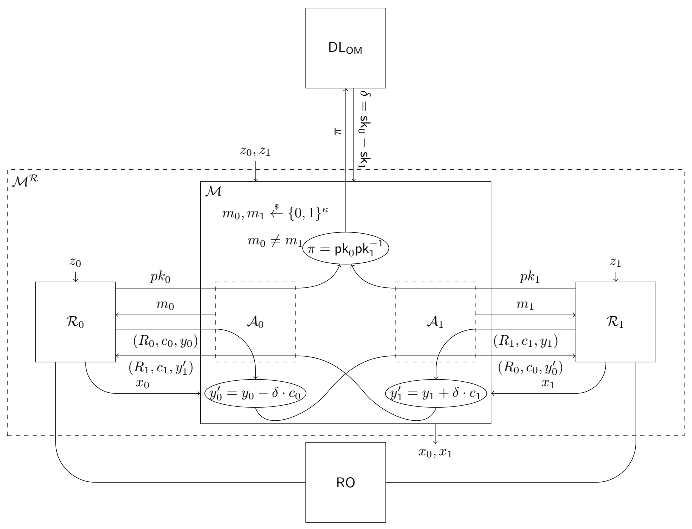
More precisely, assume there exists a non-programming single-instance fully-black-box reduction $\mathcal{R}$ that converts any adversary $\mathcal{A}$ against the $\mathsf{EUF\text{-}CMA}$ *security of* SSS *working in group $\mathbb{G}$ into an adversary against the* DL *problem over $\mathbb{G}$. Assume further that the reduction has success probability $\mathsf{Succ}^{\mathcal{R}^{\mathcal{A}}}_{\mathsf{DL},\mathbb{G}}(\kappa)$ for given $\mathcal{A}$ and runtime $\mathsf{Time}_{\mathcal{R}}(\kappa)$. Then, there exists a family $\mathbb{A}$ of successful (but possibly inefficient) adversaries $\mathcal{A}_{\mathcal{R},a}$ against the* $\mathsf{EUF\text{-}CMA}$ *security of* SSS *and a meta-reduction $\mathcal{M}$ that breaks the* $\mathsf{1\text{-}DL}$ *assumption over $\mathbb{G}$ with non-negligible success probability $\mathsf{Succ}^{\mathcal{M}}_{\mathsf{1\text{-}DL},\mathbb{G}}(\kappa) \geq (\mathsf{Succ}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R},a}}}_{\mathsf{DL},\mathbb{G}}(\kappa))^2$ for a random $\mathcal{A}_{\mathcal{R},a} \in \mathbb{A}$ and runtime $\mathsf{Time}_{\mathcal{M}}(\kappa) = 2 \cdot \mathsf{Time}_{\mathcal{R}}(\kappa) + \mathsf{poly}(\kappa)$.*

Note that the fact that $\mathcal{A}$ breaks SSS working over $\mathbb{G}$ implies that for any public key pk output by $\mathcal{R}$ it holds that $\mathsf{pk} \in \mathbb{G}$.

Proof. Roughly, the meta-reduction $\mathcal{M}$ with inputs z_0, z_1 works as follows: It invokes two instances $\mathcal{R}_0$ and $\mathcal{R}_1$ of the reduction in a black-box way, on inputs z_0 and z_1, respectively, and independent random tapes. When the instances of $\mathcal{R}$ invoke the forger with public key pk_0 and pk_1 respectively, $\mathcal{M}$ simulates a specific forger, we will describe below and which will allow the meta-reduction to be "sufficiently close" to the adversary's behavior. To do so, the meta-reduction queries random messages to the sign oracles and obtains signatures on them. It then queries the quotient of the two public keys, i.e., $\mathsf{pk}_0\mathsf{pk}_1^{-1}$, to the $\mathsf{DL}_{\mathsf{OM}}$ oracle, thus obtaining the difference between the secret keys. The difference between the secret keys can then be used to adapt the obtained signatures to the other public key, respectively. These adapted signatures are then returned to the reductions as forgeries. As we are working in the (non-programmable) random oracle model, the instances of $\mathcal{R}$ expect to see all the random oracle queries, the (simulated) adversary would issue. The meta-reduction $\mathcal{M}$ therefore makes sure to issue exactly those queries. Then the meta-reduction mimics the behavior of the adversary closely, and succeeds in solving the 1-one-more DL problem.

A Family of Adversaries $\mathcal{A}_{\mathcal{R},a}$. Let A be defined as the set $\mathsf{A} = \{0,1\}^\kappa \times \{0,1\}^\kappa \times \mathbb{G} \times \{0,1\}^{\mathsf{poly}(\kappa)}$. For every $a = (m_0, m_1, z, \varpi) \in \mathsf{A}$ and every reduction $\mathcal{R}$ we define the adversary $\mathcal{A}_{\mathcal{R},a}$ as described below. Since the reduction has to work for any adversary, it particularly needs to succeed for randomly chosen $a \leftarrow \mathsf{A}$ (and thus for randomly chosen $\mathcal{A}_{\mathcal{R},a}$ from the family), such that $\mathcal{A}_{\mathcal{R},a}$ is still deterministic but we can take the random choice of a into account when arguing about probabilities. Note we occasionally drop (parts of) the subindex from $\mathcal{A}_{\mathcal{R},a}$ for sake of simplicity.

Upon receiving as input the public key pk, adversary $\mathcal{A}_{\mathcal{R},a}$ queries m_0 to its signing oracle and verifies the reply σ_0. If the signature σ_0 is invalid, $\mathcal{A}_{\mathcal{R}}$ aborts. Otherwise, $\mathcal{A}_{\mathcal{R}}$ discards the signature and invokes an internal copy of $\mathcal{R}$ —denoted $\mathcal{R}^*$ in the following— on input z and random tape ϖ. The adversary $\mathcal{A}_{\mathcal{R}}$



$\mathsf{DL}_{\mathsf{OM}}$
π
$\delta = \mathsf{sk}_0 - \mathsf{sk}_1$
z_0, z_1
$\mathcal{M}^{\mathcal{R}}$
$\mathcal{M}$
$m_0, m_1 \xleftarrow{\$} \{0,1\}^\kappa$
$m_0 \neq m_1$
$\pi = \mathsf{pk}_0\mathsf{pk}_1^{-1}$
z_0
z_1
pk_0
pk_1
$\mathcal{R}_0$
m_0
$\mathcal{A}_0$
$\mathcal{A}_1$
m_1
$\mathcal{R}_1$
(R_0, c_0, y_0)
(R_1, c_1, y_1)
(R_1, c_1, y_1')
(R_0, c_0, y_0')
x_0
x_1
$y_0' = y_0 - \delta \cdot c_0$
$y_1' = y_1 + \delta \cdot c_1$
x_0, x_1
RO

Figure 3: The meta-reduction uses two instances of $\mathcal{R}$ and simulates the adversary $\mathcal{A}$ by obtaining the difference between the secret keys and adapting the signatures output by $\mathcal{R}$ to the other key, respectively.

forwards all random oracle queries by $\mathcal{R}^*$ to the external random oracle. When $\mathcal{R}^*$ invokes the forger by outputting a public key pk_1, $\mathcal{A}_{\mathcal{R}}$ queries m_1 to $\mathcal{R}^*$ and verifies the reply. If $\mathcal{R}^*$ is unable to sign m_1, $\mathcal{A}_{\mathcal{R}}$ aborts. Otherwise it obtains a signature $\sigma_1 = (c_1, y_1)$ on m_1, aborts $\mathcal{R}^*$, and proceeds to forge a signature.

Using exhaustive search, the adversary $\mathcal{A}_{\mathcal{R}}$ finds $\delta \in \mathbb{Z}_q$ such that $\mathsf{pk} \cdot \mathsf{pk}_1^{-1} = g^\delta$, i.e., $\delta = \mathsf{sk} - \mathsf{sk}_1 \bmod q$. Given this difference of the secret keys, it then adapts σ_1 to the public key pk by computing ${\sigma_1}^* = ({c_1}^*, {y_1}^*) := (c_1, y_1 - \delta \cdot c_1 \bmod q)$. Finally, it returns the forgery $(m_1, {\sigma_1}^*)$. Note that $\mathcal{A}_{\mathcal{R}}$'s behaviour is fully deterministic and it thus will always return the same message-signature pair, even if it is rewound. Further, observe that for any reduction $\mathcal{R}$, $\mathcal{A}_{\mathcal{R}}$ is successful with at least the same probability with which an instance of $\mathcal{R}$ is able to sign a randomly chosen messages (minus the negligible term $2^{-\kappa}$ for which the two messages collide).

As stated in the informal description of the meta-reduction above, it is necessary to consider which random oracle queries the adversary $\mathcal{A}_{\mathcal{R}}$ would issue. These queries are exactly those required for verifying σ_0, those asked by $\mathcal{R}^*$ up to the point where $\mathcal{R}^*$ answers the first signature query, and those required for verifying σ_1. These queries are all issued *after* the only signing query has already been answered. It is important that a meta-reduction simulating $\mathcal{A}_{\mathcal{R}}$ behaves in exactly the same way.

Description of $\mathcal{M}$. The meta-reduction $\mathcal{M}$, depicted in Figure 3, on input $z_0, z_1 \in \mathbb{G}$ invokes two instances of $\mathcal{R}$ with independent random tapes. The first one, in the following denoted $\mathcal{R}_0$, gets as input z_0. The second one, denoted $\mathcal{R}_1$, gets as input z_1. All random oracle queries issued by either $\mathcal{R}_0$ or $\mathcal{R}_1$ are answered by forwarding the query to the external random oracle and returning the answer. Both instances can now invoke the forger $\mathcal{A}$ at most once.

To simulate $\mathcal{A}$ for each copy, algorithm $\mathcal{M}$ now proceeds as follows:

1. Obtain $\mathsf{pk}_b \leftarrow \mathcal{R}_b$ for $b = 0, 1$ and choose $m_0, m_1 \xleftarrow{\$} \{0,1\}^\kappa$.
2. Query $(c_0, y_0) \leftarrow \mathcal{R}_0.\mathsf{Sign}(m_0)$, $(c_1, y_1) \leftarrow \mathcal{R}_1.\mathsf{Sign}(m_1)$ and verify both signatures.
3. If either $\mathcal{R}_0$ or $\mathcal{R}_1$ were unable to provide a valid signature, abort.
4. Let $\mathbb{Q}_{\mathcal{H},b}$ be the sequence of random oracle queries issued by $\mathcal{R}_b$ up to, and including, step 2. This also includes the hash queries to verify the corresponding signature.

5. Query $\mathbb{Q}_{\mathcal{H},1}$ to the random oracle interface provided by $\mathcal{R}_0$, and $\mathbb{Q}_{\mathcal{H},0}$ to the random oracle interface provided by $\mathcal{R}_1$, emulating the same hash queries the adversary instance for each reduction would issue.

6. Query $\delta \leftarrow \mathsf{DL}_{\mathsf{OM}}(\mathsf{pk}_0\mathsf{pk}_1^{-1})$ and adapt signatures: $\sigma_0' := (c_0, y_0 - \delta \cdot c_0 \bmod q)$, $\sigma_1' := (c_1, y_1 + \delta \cdot c_1 \bmod q)$.

7. Return (m_1, σ_1') as a forgery to $\mathcal{R}_0$ and (m_0, σ_0') as a forgery to $\mathcal{R}_1$.

Potentially, $\mathcal{M}$ may not obtain any signature from $\mathcal{R}_0$ (resp. $\mathcal{R}_1$) in Step 2, because the reduction has already output a solution directly. This might happen if the instance either never invokes the adversary or chooses not to answer the sign query and directly outputs the solution instead. If this happens, then the meta-reduction would have received one of the discrete logarithms without invoking its $\mathsf{DL}_{\mathsf{OM}}$ oracle. Therefore, it could just abort the other reduction instance, query the remaining input to the $\mathsf{DL}_{\mathsf{OM}}$ oracle and output the solutions. In the following we can therefore assume that the reductions actually run an instance of $\mathcal{A}$ and answer the signature queries.

If $\mathcal{R}_b$ for $b \in \{0,1\}$ tries to rewind the forger, $\mathcal{M}$ will keep querying m_b, issuing $\mathbb{Q}_{\mathcal{H},1-b}$ as random oracle queries, and outputting (m_{1-b}, σ_{1-b}') as a forgery. Note that this behavior is consistent with the behavior of $\mathcal{A}_{\mathcal{R},(m_b, m1-b, x_{1-b}, \varpi_{1-b})}$, where ϖ_{1-b} is the random tape used by $\mathcal{R}_{1-b}$. After both instances of $\mathcal{R}$ have invoked at most one instance of $\mathcal{A}$ and possibly rewound a polynomial number of times, each outputs a solution candidate x_0 and x_1, respectively. Finally, $\mathcal{M}$ outputs x_0, x_1.

Observe that the number of $\mathsf{DL}_{\mathsf{OM}}$ oracle queries is independent of the forger. In fact, $\mathcal{M}$ queries the $\mathsf{DL}_{\mathsf{OM}}$ exactly once to simulate both forgers. Therefore, $\mathcal{M}$ is a valid adversary in the 1-DL experiment. This is also the point where it is crucial that $\mathcal{R}$ is *single-instance*, see Remark 3.2 below.

The runtime of $\mathcal{M}$ consists of two executions of $\mathcal{R}$, n oracle calls, and a constant number of modular inversions, multiplications, and additions. Therefore, it holds that $\mathsf{Time}_{\mathcal{M}}(\kappa) = 2 \cdot \mathsf{Time}_{\mathcal{R}}(\kappa) + \mathsf{poly}(n)$. Furthermore, whenever both $\mathcal{R}_0$ and $\mathcal{R}_1$ are successful, i.e., they both managed to compute the discrete logarithm of their input, $\mathcal{M}$ knows the discrete logarithms of all its inputs and is, therefore, also successful. However, the question remains if the simulation of $\mathcal{A}$ is convincing, i.e., indistinguishable from the real forger.

Correctness of the Forger Simulation. It is easy to see that the adaption used by $\mathcal{M}$ yields valid signatures under the correct public key:

$$y_0' := y_0 - \delta \cdot c_0 = r_0 + \mathsf{sk}_0 \cdot c_0 - (\mathsf{sk}_0 - \mathsf{sk}_1) \cdot c_0 = r_0 + \mathsf{sk}_1 \cdot c_0 \bmod q$$
$$y_1' := y_1 + \delta \cdot c_1 = r_1 + \mathsf{sk}_1 \cdot c_1 + (\mathsf{sk}_1 - \mathsf{sk}_0) \cdot c_1 = r_1 + \mathsf{sk}_0 \cdot c_1 \bmod q$$

The question remains, whether the forgeries presented by $\mathcal{M}$ —coming from instances of the reduction itself— might have some structure that makes them useless to the reduction, thus lowering the reduction's success probability and causing $\mathcal{M}$ to fail. Fortunately, this is not the case, because $\mathcal{M}$ perfectly mimics $\mathcal{A}_{\mathcal{R},a}$ for some $a \in \mathsf{A}$. Both algorithms compute the forgery as follows: Run an instance of $\mathcal{R}$ on a random input and an independent random tape. Obtain a signature on a randomly chosen message m from said instance. Adapt the signature to the public key received as input using the difference of the secret keys. Thus, the forgeries are distributed identically.

Furthermore, the output behavior of both algorithms is identical. Both algorithms, on input a public key, first query a random message to the signing oracle. Then, they issue exactly those random oracle queries needed to verify the received signature, and the ones asked by $\mathcal{R}$ up to the point where $\mathcal{R}$ answers the first signing query. Finally, they both output the forgery, which, as mentioned above, is distributed identically in both cases. When rewound, the outputs of both algorithms remain the same.

Therefore, $\mathcal{A}_{\mathcal{R}}$ and $\mathcal{M}$ are perfectly indistinguishable from $\mathcal{R}$'s point of view and thus $\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{M}}}(\kappa) = \mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R}}}}(\kappa)$. As stated before, $\mathcal{M}$ is successful whenever both instances of the reduction are successful and, therefore, $\mathsf{Succ}_{1\text{-}\mathsf{DL}}^{\mathcal{M}}(\kappa) = (\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{M}}}(\kappa))^2 = (\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R}}}}(\kappa))^2$.

It remains to show that $\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R},a}}}(\kappa)$ is non-negligible. To this end, consider the following (all-powerful) adversary $\mathcal{A}_0$ against the signature scheme which asks for a signature for a random message $m \in \{0,1\}^\kappa$ (which remains unchanged after resets). If this adversary receives a valid signature, it exhaustively searches for the secret key and then creates a signature for another random but fixed message $m^* \in \{0,1\}^\kappa$. This adversary succeeds with probability $1-2^{-\kappa}$ if attacking the actual scheme, taking into account the probability $2^{-\kappa}$ of a match of the two messages. Hence, the reduction $\mathcal{R}$ here must succeed with some non-negligible probability if given this (black-box) adversary.

We first assume that the probability that the reduction $\mathcal{R}_0$ (the same argument holds for $\mathcal{R}_1$) succeeds in computing the discrete logarithm of z_0 and does not answer $\mathcal{A}_0$'s (unique) signature query, i.e., $\mathcal{R}_0$ succeeds without $\mathcal{A}_0$'s help, is negligible. Otherwise, if it was non-negligible, our meta-reduction would already succeed with non-negligible probability, as the behavior of $\mathcal{A}_\mathcal{R}$ and of $\mathcal{A}_0$ are identical, and our meta-reductions wins for such reductions. We may therefore assume that $\mathcal{R}_0$ must correctly answer the only signature query of $\mathcal{A}_0$ with some non-negligible probability. But then this is also true for our adversary $\mathcal{A}_\mathcal{R}$ and the internal copy $\mathcal{R}^*$, as both reductions are initialized randomly and, up to the signature query, the reductions' views for both adversaries are identical. Given that our adversary $\mathcal{A}_{\mathcal{R},a}$ here receives a valid signature from its internal copy $\mathcal{R}^*$, it succeeds with probability $1-2^{-\kappa}$. Therefore, the overall success probability of $\mathcal{A}_{\mathcal{R},a}$ is non-negligible. But then $\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R},a}}}(\kappa)$ must be non-negligible, too, for a successful reduction. This completes the final step in the proof. □

Remark. Note that the restriction to single-instance reductions is crucial at this point. Consider a reduction that would output a second public key, either by invoking another instance of $\mathcal{A}$ or by rewinding the adversary to a point before it received the public key. The meta-reduction would then need to issue another query to the $\mathsf{DL}_{\mathsf{OM}}$ oracle to simulate the signing oracle. Obviously, $\mathcal{M}$ would then have made 2 queries to the $\mathsf{DL}_{\mathsf{OM}}$ oracle and could, thus, no longer win in the 1-DL experiment.

Remark. While it may seem strange at first that the adversary $\mathcal{A}_{\mathcal{R},a}$ is defined in terms of the reduction, remember that we are talking about fully black-box reductions here. A reduction $\mathcal{R}$ needs to work for any adversary against the signature scheme. In particular it needs to work for the adversaries $\mathcal{A}_{\mathcal{R},a}$ for any $a \in \mathbb{A}$.

Remark. It should be noted, that the meta-reduction employed in the proof of Theorem 3.4 only works because SSS is defined relative to a single fixed random oracle. If one uses a common variant of the Fiat-Shamir transform, in which the random oracle is "personalized" by including the public key in the hash query, $c = H(\mathsf{pk}, R, m)$, the meta-reduction no longer works. This is due to the fact that in this case signatures can no longer be simply adapted to another public key, using only the secret keys' difference.

Remark. The idea immediately applies to other FS signature schemes with unique keys, where there is a related one-more problem, such as the RSA-GQ scheme [GQ88].

4 Limitations of the Meta-Reduction Technique

Paillier and Vergnaud [PV05], as well as we here, have used meta-reductions to provide evidence that, once we drop programmability, the security of Schnorr signatures might not be equivalent to the discrete log problem after all. Both results are not definitive proofs insofar as they restrict the reduction's abilities. Paillier and Vergnaud rule out algebraic reductions in the standard model. We rule out non-programming single-instance reductions which use the adversary only in a black-box way. There could in principle still exist a reduction that does not fall into either of these classes. However, it is interesting to note that in both cases the meta-reduction-based proofs rely on the one-more discrete log assumption. As the discrete log assumption does not seem to imply its one-more variants [BMV08] the results are, thus, conditional and not as strong as they could be. The obvious question is therefore: "Can we do better?" Unfortunately, the answer turns out to be "Not without finding an actual adversary."

Our results actually holds for *any* randomized signature scheme $\mathcal{S}$ (where, as explained in Section 2, the signing algorithm has super-logarithmic min entropy) for which the signing algorithm's hash queries in any signature generation can always be reconstructed from the signature alone, in the right order. We call them randomized signature schemes *with reconstructible hash queries*, refer to Definition 2.6 for a formal definition. These schemes include Fiat-Shamir transformed schemes such as Schnorr but also cover (randomized versions of) FDH-RSA signatures. We show that finding a meta-reduction to a non-interactive problem such as the discrete log problem is at least as hard as finding an adversary against the strong existential unforgeability of $\mathcal{S}$. For this, we first describe an inefficient reduction $\mathcal{R}$ that is capable of detecting when the forgery it receives is actually one of the signatures it produced itself as an answer to a signing query. For example, a meta-reduction may make several signing requests to a reduction and then reset these requests in order to use one additional message-signature pairs as the forgery. Our reduction will be able to spot such attempts.

The meta-reduction result of the previous section does not apply here, even though our reduction here will be of the single-instance type. The reason is that the meta-reduction there assumed an (interactive) one-more DL problem –and made use of the DL oracle– whereas the meta-reduction here should work for non-interactive problems such as the discrete log problem.

4.1 An Inefficient Reduction for Randomized Signature Schemes with Reconstructible Hash Queries

Let $\mathcal{S}$ be a randomized signature scheme and let $\Pi_{\mathcal{R}}$ be a monotone solvable problem. Let $\mathbb{Q}$ be the set of message-signature pairs (m_i, σ_i) resulting from queries to $\mathcal{R}$'s signing oracle. Furthermore, let p be the maximum number of signature queries issued by a forger $\mathcal{A}$ and assume that $\mathcal{R}$ knows the polynomially bounded p. We note that for the adversary in our single-instance reductions in the previous section, the reduction could have been given $p = 1$, too. For the moment, the reader may think of the meta-reduction as running a single instance of the reduction; we will later reduce the multi-instance case to the single-instance case via standard "guess-and-insert" techniques.

The reduction $\mathcal{R}$, depicted in Figure 4, on input an instance z of $\Pi_{\mathcal{R}}$ first generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$, then initializes the counter variable $i := 0$, and chooses a random function $\mathcal{O} : \{0,1\}^{2^\kappa} \times \mathbb{Z}_p \rightarrow \mathit{Coins}_{\mathsf{pk}}$. The public key pk is then output as the key under which the forger is supposed to forge a signature. When the forger queries a message m to the signing oracle, $\mathcal{R}$ determines random coins

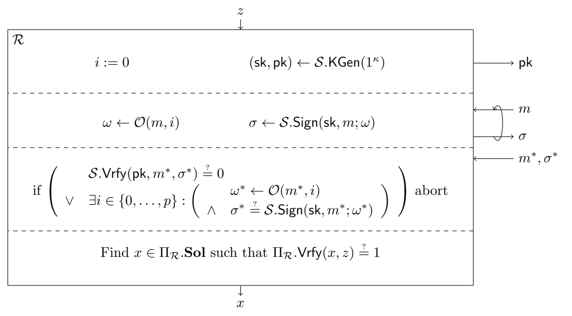

z
$\mathcal{R}$
$i := 0$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$
pk
m
$\omega \leftarrow \mathcal{O}(m, i)$
$\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m; \omega)$
σ
m^*, σ^*
if $\left(\begin{array}{l} \mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \overset{?}{=} 0 \\ \vee \quad \exists i \in \{0, \ldots, p\} : \left(\begin{array}{l} \omega^* \leftarrow \mathcal{O}(m^*, i) \\ \wedge \quad \sigma^* \overset{?}{=} \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*; \omega^*) \end{array} \right) \end{array} \right)$ abort
Find $x \in \Pi_{\mathcal{R}}.\mathbf{Sol}$ such that $\Pi_{\mathcal{R}}.\mathsf{Vrfy}(x, z) \overset{?}{=} 1$
x

Figure 4: The reduction uses a random function $\mathcal{O}$ to decide whether it is being re-fed its own output. This way it can abort if it is used in such a way by a potential meta-reduction.

$\omega \leftarrow \mathcal{O}(m, i)$, computes the signature as $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m; \omega)$, and returns σ to the forger. The counter i is then incremented by one. If the counter is ever incremented to $p + 1$, then $\mathcal{R}$ aborts, as it is obviously not interacting with the real adversary.

Eventually, the forger outputs a forgery (m^*, σ^*). If the signature does not verify, i.e., $\mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \overset{?}{=} 0$, then $\mathcal{R}$ immediately aborts. Otherwise, the reduction computes $\sigma_j \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*; \omega_j)$ with $\omega_j \leftarrow$

$\mathcal{O}(m^*, j)$ for all $j \in \mathbb{Z}_p$ and checks whether $\sigma_j \stackrel{?}{=} \sigma^*$. If the check holds for any σ_j, then $\mathcal{R}$ also immediately aborts. Otherwise, $\mathcal{R}$ enumerates all possible solutions $x \in \Pi_{\mathcal{R}}.\mathbf{Sol}$ and checks whether $\Pi_{\mathcal{R}}.\mathsf{Vrfy}(x, z) \stackrel{?}{=} 1$. Once such an x is found, it is output by $\mathcal{R}$ as the solution. Because $\Pi_{\mathcal{R}}$ is monotone and solvable, it is guaranteed that there exists a valid solution even though $\mathcal{R}$ never issues a single oracle query and that the enumeration of possible solutions will terminate in finite time.

Observe that the adversary $\mathcal{A}$ used by $\mathcal{R}$ is an $\mathsf{EUF\text{-}CMA}$ adversary, therefore, whenever $\mathcal{A}$ forges successfully, it forges a signature for a message m^* that has not been queried before. The probability that $\mathcal{R}$ will reject such a forgery is the probability that at least one of the σ_i collides with σ^*. As $\mathcal{O}$ is a random function, all values to which $\mathcal{O}$ evaluates on input m^* and some number i are uniformly and independently distributed. For each σ_i, the probability that it matches σ^* is thus bounded through the min-entropy of the random variable describing $\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*)$, i.e., $\forall i \in \mathbb{Z}_p : (\Pr[\sigma_i \stackrel{?}{=} \sigma^*] \leq 2^{-H_\infty(\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*))})$.

Therefore, the probability that $\mathcal{R}$ will accept a forgery is at least $1 - p \cdot 2^{-H_\infty(\mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m^*))}$. As $\mathcal{S}$ is randomized, the probability for each σ_i to match is negligible and thus

$$\mathsf{Succ}_{\Pi_{\mathcal{R}}}^{\mathcal{R}^{\mathcal{A}}}(\kappa) \geq (1 - p \cdot \epsilon(\kappa)) \cdot \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) = \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) - \epsilon'(\kappa)$$

for negligible functions ϵ, ϵ'. Therefore, we conclude that $\mathsf{Succ}_{\Pi}^{\mathcal{R}^{\mathcal{A}}}(\kappa)$ is non-negligible for any successful adversary $\mathcal{A}$ and that $\mathcal{R}$ is, thus, a successful –albeit inefficient– reduction from problem $\Pi_{\mathcal{R}}$ to the $\mathsf{EUF\text{-}CMA}$ security of $\mathcal{S}$.

We next show that the checks of our reduction prevent the meta-reduction to replay signatures to the reduction. This step relies on the fact that the meta-reduction can only use the reduction in a black-box way, $\mathcal{M}^{\mathcal{R}}$, and has for example no control over the coin tosses of $\mathcal{R}$. First, we show that we can restrict ourselves to meta-reductions which actually take advantage of the reduction, at least if the meta-reduction's problem $\Pi_{\mathcal{M}}$ is hard:

Lemma 4.1 (Meta-Reductions Rely on the Reduction) *Let $\mathcal{M}$ be a non-programming meta-reduction that converts any ($\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}}$) reduction in a black-box way into an adversary against some hard problem $\Pi_{\mathcal{M}}$. Further, let the reduction used by $\mathcal{M}$ be $\mathcal{R}$ as described above. Then it holds that $\mathcal{M}$ provides $\mathcal{R}$ with a forgery (m^*, σ^*) with non-negligible advantage.*

Proof. Assume that this was not the case. Then one could easily simulate $\mathcal{R}$ and the meta-reduction interacting with this reduction would solve $\Pi_{\mathcal{M}}$ efficiently with non-negligible advantage. This contradicts the hardness of the problem. □

Hence, from now on we condition on the meta-reduction to always provide the reduction with a forgery, without losing more than a negligible advantage. In this case we have:

Lemma 4.2 (Meta-Reductions Cannot Replay Signatures) *Let $\mathcal{M}$ be a non-programming meta-reduction that converts any ($\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}}$) reduction in a black-box way into an adversary against some hard problem $\Pi_{\mathcal{M}}$. Let $\mathbb{Q}$ be the set of message-signature pairs (m_i, σ_i) resulting from $\mathcal{M}$'s queries to the reduction's signing oracle, and let (m^*, σ^*) be the message-signature pair output by $\mathcal{M}$ as a forgery on behalf of the adversary. Further, let the reduction used by $\mathcal{M}$ be $\mathcal{R}$ as described above. Then it holds that $(m^*, \sigma^*) \notin \mathbb{Q}$.*

Proof. The proof is rather straightforward. Observe that by construction of $\mathcal{R}$ the following holds: $\forall (m, \sigma) \in \mathbb{Q} : \exists i \in \mathbb{Z}_p : \omega \leftarrow \mathcal{O}(m, i) \wedge \sigma \stackrel{?}{=} \mathcal{S}.\mathsf{Sign}(m, i; \omega)$. Therefore, it follows directly that, for $(m^*, \sigma^*) \in \mathbb{Q}$, the reduction $\mathcal{R}$ will abort and $\mathsf{Succ}_{\Pi_{\mathcal{R}}}^{\mathcal{R}^{\mathcal{M}}}(\kappa) = 0$ for $\mathcal{M}$ if it replays an element of $\mathbb{Q}$ as a forgery. Note that here we rely on the previous Lemma which assumes that $\mathcal{M}$ always provides such a forgery. As it, thus, would not be a successful meta-reduction it must hold that $(m^*, \sigma^*) \notin \mathbb{Q}$. □

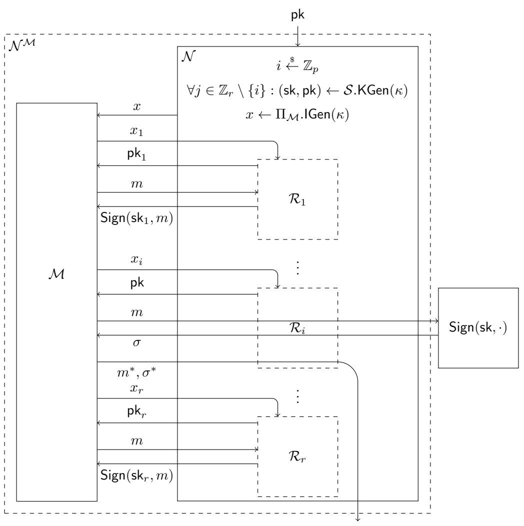
4.2 A Reduction against the Meta-Reduction

Using the reduction described in the previous section, we now prove that finding an efficient meta-reduction for a randomized signature scheme is at least as hard as finding a strong existential forger.

Theorem 4.3 (Meta-Reductions to Non-Interactive Problems Are Hard) *Let $\mathcal{S}$ be a randomized signature scheme with reconstructible hash queries, $\Pi_{\mathcal{R}}$ be a monotone solvable problem, and $\Pi_{\mathcal{M}}$ be a non-interactive, efficiently generatable problem. If $\Pi_{\mathcal{M}}$ is hard, then finding an efficient meta-reduction $\mathcal{M}$ that converts any successful $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$-reduction in a black-box way into an efficient successful adversary against $\Pi_{\mathcal{M}}$ is at least as hard as finding an* sEUF-CMA *adversary against $\mathcal{S}$.*

More precisely, assume there exists an efficient non-programming black-box meta-reduction $\mathcal{M}$ that converts any $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_{\mathcal{R}})$-reduction into an adversary against $\Pi_{\mathcal{M}}$. Then, there exists a meta-meta-reduction $\mathcal{N}$ that converts $\mathcal{M}$ into an adversary against the sEUF-CMA *security of $\mathcal{S}$ with non-negligible success probability $\mathsf{Succ}^{\mathcal{S},\mathcal{N}^{\mathcal{M}}}_{\mathsf{sEUF\text{-}CMA}}(\kappa) \geq \frac{1}{r} \cdot \mathsf{Succ}^{\mathcal{M}^{\mathcal{R}}}_{\Pi_{\mathcal{M}}}(\kappa)$ and runtime $\mathsf{Time}_{\mathcal{N}^{\mathcal{M}}}(\kappa) = \mathsf{Time}_{\mathcal{M}^{\mathcal{R}}}(\kappa) + \mathsf{poly}(\kappa)$, where $\mathcal{R}$ is the reduction described above and r is the maximal number of reduction instances invoked by $\mathcal{M}$.*

Note that, since $\mathcal{M}$ needs to work for any (black-box) $\mathcal{R}$, we may assume that $\mathcal{R}$ knows r. Indeed, we take advantage of this fact in the proof below.



pk
$\mathcal{N}^{\mathcal{M}}$
$\mathcal{N}$
$i \xleftarrow{\$} \mathbb{Z}_p$
$\forall j \in \mathbb{Z}_r \setminus \{i\} : (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(\kappa)$
$x \leftarrow \Pi_{\mathcal{M}}.\mathsf{IGen}(\kappa)$
x
x_1
pk_1
m
$\mathcal{R}_1$
$\mathsf{Sign}(\mathsf{sk}_1, m)$
$\mathcal{M}$
x_i
pk
m
$\mathcal{R}_i$
σ
$\mathsf{Sign}(\mathsf{sk}, \cdot)$
m^*, σ^*
x_r
pk_r
m
$\mathcal{R}_r$
$\mathsf{Sign}(\mathsf{sk}_r, m)$

Figure 5: The meta-meta-reduction relies on the fact that $\mathcal{M}$ cannot replay an old signature. It guesses the reduction instance, for which $\mathcal{M}$ will output a forgery, embeds the public key in that instance, and outputs the forgery provided by $\mathcal{M}$.

Proof. We construct $\mathcal{N}$ –depicted in Figure 5– such that, from $\mathcal{M}$'s perspective, it is perfectly indistinguishable from the reduction $\mathcal{R}$. On input a public key pk of the "outer" signature scheme $\mathcal{S}$ the meta-meta-reduction $\mathcal{N}$ chooses a random $i \xleftarrow{\$} \mathbb{Z}_r$ to guess the "good" instance among the r copies of the reduction, and sets $\mathsf{pk}_i = \mathsf{pk}$. For all $j \in \mathbb{Z}_r \setminus \{i\}$, algorithm $\mathcal{N}$ computes $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathsf{KGen}(1^\kappa)$. Then it generates a random instance $x \leftarrow \Pi_{\mathcal{M}}.\mathsf{IGen}(1^\kappa)$ and invokes $\mathcal{M}$ on x.

When $\mathcal{M}$ invokes the jth instance of $\mathcal{R}$ on an instance x_j of $\Pi_{\mathcal{R}}$, $\mathcal{N}$ outputs pk_j for $\mathcal{M}$ to forge a signature under. Queries to sign message m under public key pk_j are answered differently depending on whether $j = i$ or $j \neq i$. If $j = i$, m is simply relayed to $\mathcal{N}$'s own signature oracle and the answer is sent back to $\mathcal{M}$. The meta-meta-reduction also makes the hash queries the signature algorithm has made in the right order; this is possible by the reconstruction property. If $j \neq i$, $\mathcal{N}$ computes $\sigma \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}_j, m)$ and sends σ back to $\mathcal{M}$. To be consistent with the behavior of $\mathcal{R}$, it is important to handle rewinding of the simulated reductions correctly. In particular, $\mathcal{R}$ can be rewound to a previous state, where the state basically consists of the value of its counter variable. If $\mathcal{R}$ is now queried some message m which it had already been queried before in the same state, it will respond with exactly the same random oracle queries and the same signature on the message. If, on the other hand, the message m has not been queried before in the same state, $\mathcal{R}$'s response is based on completely independent coins. To mimic this behavior, for every simulated reduction instance, $\mathcal{N}$ keeps track of the value the counter variable would currently have in an actual instance of $\mathcal{R}$. For each counter value it then stores all triples (m, σ, Q) of messages, resulting signature, and queries issued to the random oracle during computation of σ that already resulted in that state. If one of those messages is queried again, $\mathcal{N}$ issues Q to the random oracle again and returns σ.

The random oracle queries required for simulating the reduction instances are all issued to the random oracle interface provided by the meta-reduction. Queries of the meta-reduction to the random oracle on the other hand are honestly answered by consulting the external random oracle. Note that the query of the external signature oracle to the random oracle can be made by $\mathcal{N}$ itself after receiving the signature. Eventually, $\mathcal{M}$ outputs the first forgery (m^*, σ^*) under some public key pk_j. If $j \neq i$, $\mathcal{N}$ aborts. If $j = i$, $\mathcal{N}$ stops the execution of $\mathcal{M}$ and outputs (m^*, σ^*).

Correctness of Reduction Simulation. We show that $\mathcal{N}$ faithfully simulates a maximum of r instances of $\mathcal{R}$. The public key in the $\mathsf{sEUF\text{-}CMA}$ game, as well as those chosen by $\mathcal{N}$ itself are honestly computed using $\mathcal{S}$'s key generation algorithm. Therefore the public keys output by $\mathcal{N}$ are distributed identically to keys output by several instances of $\mathcal{R}$. The same holds for the answers to the signature queries, as in both cases the signatures are honestly computed using uniformly and independently chosen random coins and the random oracle queries issued are exactly those required for an honest execution of $\mathcal{S}.\mathsf{Sign}$. Further when $\mathcal{M}$ tries to rewind the reduction, both $\mathcal{R}$ and the simulation by $\mathcal{N}$ behave identically. Therefore, $\mathcal{N}$ faithfully simulates r instances of $\mathcal{R}$ up to the point where $\mathcal{M}$ outputs the first forgery.

As the behavior of $\mathcal{N}$ is perfectly indistinguishable from the behavior of a number of $\mathcal{R}$ instances, we can now apply Lemma 4.2 (which, in turn, relies on Lemma 4.1). We thus conclude that, if $\mathcal{M}$ would have been successful, it holds that (m^*, σ^*) is a valid message-signature pair and that $(m^*, \sigma^*) \notin \mathbb{Q}$. If $i = j$, these are exactly the conditions for $\mathcal{N}$ to win in the $\mathsf{sEUF\text{-}CMA}$ game. As i is chosen uniformly and $\mathcal{M}$ is perfectly oblivious about i, we get $\mathsf{Succ}^{\mathcal{S}, \mathcal{N}^{\mathcal{M}}}_{\mathsf{sEUF\text{-}CMA}}(\kappa) \geq \frac{1}{r} \cdot \mathsf{Succ}^{\mathcal{M}^{\mathcal{R}}}_{\Pi_{\mathcal{M}}}(\kappa)$ as claimed. Besides executing $\mathcal{M}$ the meta-meta-reduction $\mathcal{N}$ needs to generate an instance of $\Pi_{\mathcal{M}}$, generate $r-1$ key pairs, and answer a polynomial number of sign queries. All of this is possible in polynomial time, because $\Pi_{\mathcal{M}}$ is efficiently generatable and the algorithms of $\mathcal{S}$ obviously need to be efficient. Therefore, $\mathsf{Time}_{\mathcal{N}^{\mathcal{M}}}(\kappa) = \mathsf{Time}_{\mathcal{M}^{\mathcal{R}}}(\kappa) + \mathsf{poly}(\kappa)$ as claimed. □

Acknowledgments

We thank the anonymous reviewers for valuable comments. Marc Fischlin was supported by a Heisenberg grant Fi 940/3-1 of the German Research Foundation (DFG). Part of this work was also supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE. Nils Fleischhacker was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA – `www.cispa-security.org`).

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Berlin, Germany.

[AGO11] Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 628–646, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Germany.

[BBF13] Paul Baecher, Christina Brzuska, and Marc Fischlin. Notions of black-box reductions, revisited. *IACR Cryptology ePrint Archive*, 2013.

[BL12] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. *IACR Cryptology ePrint Archive*, 2012.

[BMV08] Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the “one-more” computational problems. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 71–87, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Germany.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Berlin, Germany.

[FF02] Marc Fischlin and Roger Fischlin. The representation problem based on factoring. In Bart Preneel, editor, *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 96–113, San Jose, CA, USA, February 18–22, 2002. Springer, Berlin, Germany.

[Fis12] Marc Fischlin. Black-box reductions and separations in cryptography. In *AFRICACRYPT*, volume 7374 of *Lecture Notes in Computer Science*, pages 413–422. Springer, 2012.

[FLR$^+$10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 303–320, Singapore, December 5–9, 2010. Springer, Berlin, Germany.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Germany.

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 197–215, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[GBL08] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 93–107, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[GQ88] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory. In C. G. Günther, editor, *Advances in Cryptology – EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Davos, Switzerland, May 25–27, 1988. Springer, Berlin, Germany.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, San Jose, California, USA, June 6–8, 2011. ACM Press.

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Berlin, Germany.

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 109–118, San Jose, California, USA, June 6–8, 2011. ACM Press.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 1–20, Chennai, India, December 4–8, 2005. Springer, Berlin, Germany.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Seu12] Yannick Seurin. On the exact security of schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 554–571, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.