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

Grenzen der Metareduktionstechnik: Der Fall von Schnorr Signaturen

Master Thesis by Nils Fleischhacker

September 25, 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

Cryptoplexity

Cryptography & Complexity Theory
Technische Universität Darmstadt
www.cryptoplexity.de

Limitations of the Meta-Reduction Technique:
The Case of Schnorr Signatures
Grenzen der Metareduktionstechnik: Der Fall von Schnorr Signaturen

Master Thesis by Nils Fleischhacker

Advisor: Prof. Dr. Marc Fischlin

For all their love,
for their endless patience,
for their never wavering support,
this is for my parents.

Erklärung zur Master Thesis

Hiermit versichere ich, die vorliegende Master Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 25. September 2012

(Nils Fleischhacker)

Abstract

Our results are twofold. On the one hand, we use the meta-reduction technique to show that the security of Schnorr signatures cannot be proven equivalent to the discrete logarithm problem without programming the random oracle. This result holds under the one-more discrete logarithm assumption and for a large natural class of reductions, we call *simple* reductions. This class in particular subsumes those used in previous proofs of security in the (programmable) random oracle model.

On the other hand, we show that a stronger result, i.e., an unconditional one, where the meta-reduction also solves the discrete logarithm problem, is most likely not achievable. In fact, we prove that for any randomized signature scheme, finding a meta-reduction that solves a hard non-interactive problem is equivalent to finding an adversary against the strong existential unforgeability of the signature scheme.

Zusammenfassung

Unsere Ergebnisse gehen in zwei Richtungen. Einerseits zeigen wir mittels Metareduktionstechnik, dass die Sicherheit von Schnorr-Signaturen nicht als äquivalent zum diskreten Logarithmus-Problem bewiesen werden kann, ohne das Random-Oracle zu programmieren. Dieses Ergebnis basiert auf der One-More-Discrete-Logarithm Annahme und gilt für eine große natürliche Klasse von Reduktionen, die wir als *einfach* bezeichnen. Insbesondere beinhaltet diese Klasse die Reduktionen, welche für bisherige Sicherheitsbeweise im (programmierbaren) Random Oracle Modell benutzt wurden.

Andererseits zeigen wir, dass es unwahrscheinlich ist ein stärkeres Resultat zu beweisen, in dem die Metareduktion das diskrete Logarithmus-Problem bricht und so ohne weitere Annahmen auskommt. Tatsächlich zeigen wir für jedes randomisierte Signaturverfahren, dass eine Metareduktion, die ein hartes, nicht-interaktives Problem löst, mindestens genauso schwer zu finden ist, wie ein tatsächlicher Angreifer gegen die starke Unfälschbarkeit des Signaturverfahrens.

Contents

List of Figures

1 Introduction

1.1 The Meta-Reduction Technique

Meta-reductions are a technique for obtaining black-box separation results. The concept was first introduced in [6], though the term *meta-reduction* was not used yet. The idea is similar to that of a normal security reduction but works against a reduction instead of an adversary, which is why it is often described as a "reduction against the reduction".

A security reduction $\mathcal{R}$ is usually given black-box access to an adversary $\mathcal{A}$ which supposedly attacks some cryptographic scheme $\mathcal{S}$. However, the scheme $\mathcal{S}$ is now simulated by $\mathcal{R}$ and $\mathcal{R}^{\mathcal{A}}$ solves some problem Π. If Π is hard and $\mathcal{R}$ is efficient, then this implies that no efficient $\mathcal{A}$ can exist.

Now, in the case of a meta-reduction $\mathcal{M}$, $\mathcal{M}$ is given black-box access to the reduction $\mathcal{R}$ and simulates the adversary $\mathcal{A}$ as well as the interaction with Π. The algorithm $\mathcal{M}^{\mathcal{R}}$ solves some problem Π' – with possibly $\Pi' = \Pi$ – and does not assume that an efficient $\mathcal{A}$ actually exists.

If Π' is hard and $\mathcal{M}$ is efficient, then this implies that no efficient $\mathcal{R}$ can exist. I.e., a black-box separation between scheme $\mathcal{S}$ and problem Π has been established under an assumption on the hardness of Π'.

If $\Pi' = \Pi$, the implication is even stronger. In this case, there can only exist a reduction from Π to the security of scheme $\mathcal{S}$ if Π is easy and therefore, if a reduction would exist, it would be useless.

In recent years, the technique has gained quite some attention and was used to achieve a number of different impossibility results, such as [8, 15, 11, 16, 12, 21, 2].

The meta-reduction technique is especially well suited to obtain separations of concrete primitives from hardness assumptions, because the construction of the primitive does not have to be black-box.

1.2 The Random Oracle Model

In the random oracle model (ROM) [5] all parties are given oracle access to a random function, called the random oracle (RO). The random oracle is often used as an idealization of a cryptographic hash function. The model allows for the construction of elegant, efficient, and provably secure cryptographic primitives and protocols, such as digital signatures. However, it has been shown that proofs of security in the ROM do not necessarily translate to security in the standard model.

In particular, Abdalla et al. [1] showed that the Fiat-Shamir paradigm [9] produces signature schemes provably secure in the ROM. Nevertheless, Goldwasser and Taumann Kalai [13] could show that in the standard model, the resulting signature schemes may be insecure regardless of how the random oracle is instantiated. This possible discrepancy is due to the fact that reductions in the ROM may exploit a number of the random oracle's properties that are impossible or at least very unlikely to be realized in the standard model.

Besides the obvious property — perfectly uniform distribution of the output — there are other properties that seem to be more subtle. One of these properties is *programmability*, the ability of the security reduction to dynamically choose the range points of the random oracle. This ability is in fact crucial for many well known security reductions in the ROM. The role of programmability in security proofs was first investigated by Nielsen [14]. Fischlin et al. [10] investigated the role of programmability further and defined three different flavors of the ROM with different levels of programmability – full, limited, and none. With full programmability the reduction can dynamically embed values of its own choice as range points of the random oracle. With limited programmability the reduction can still change the random oracle's range points dynamically, however, it is no longer able to choose them freely. With no programmability the random oracle is a fixed random function and the reduction cannot change it's output dynamically. Among other results, Fischlin et al. show that any reduction of the security of the FDH signature scheme [5] must rely on the full programmability of the random oracle.

The proof by Goldwasser and Taumann Kalai mentioned above is of course concerning. This is especially the case, once one considers the large number of signature schemes constructed using the Fiat-Shamir paradigm. However, despite their proof and despite such results as [10], to this date, there exists not a single successful attack against a "real world" Fiat-Shamir transformed signature scheme. I.e., an attack against a scheme not explicitly constructed to be susceptible to attacks in the standard model. But as the schemes also do not have a proof of security, once the random oracle has been instantiated, the question whether they are secure in the standard model remains an open one.

1.3 Schnorr Signatures

The Schnorr identification scheme was introduced by Schnorr [19, 20] as a protocol to prove knowledge of the discrete logarithm of a publicly known group element. It is a simple two-party protocol between a prover P and a verifier V that consists of three rounds. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order q. By engaging in the protocol, P can prove to V that it knows the discrete logarithm x of some public group element $z = g^x$. In the commitment round, P selects $r \xleftarrow{\$} \mathbb{Z}_q$, computes $R := g^r$, and sends R to V. In the challenge round, V selects $c \xleftarrow{\$} \mathbb{Z}_q$ and sends c to P. In the response round, P responds with $y := r + xc \mod q$. Finally, V verifies whether $R \stackrel{?}{=} z^{-c} g^y$ holds and acknowledges that P knows x if the equality holds.

By applying the Fiat-Shamir transform [9], the Schnorr signature scheme (SSS) is derived from the Schnorr identification scheme. It has been shown by Abdalla et al. [1] that, provided that the identification protocol meets certain conditions, the signature scheme resulting from the Fiat-Shamir transform is secure in the ROM. The Schnorr identification scheme can also be transformed into a blind signature scheme in a similar fashion. However, Baldimtsi and Lysyanskaya [3] recently showed that even in the ROM, basically all known techniques for proving the security of this blind variant will fail.

The specific security of SSS has been analyzed in the ROM by Pointcheval and Stern [17, 18]. They give a reduction based on the so-called Forking Lemma that basically works by rewinding the forger and reprogramming the random oracle between forger runs until two signatures are obtained that are related in a specific way. The reduction is quite loose, as is loses a factor of $\mathcal{O}(q_H)$, where q_H is the maximum number of random oracle queries by the adversary. As recently argued by Seurin [21], it is unlikely that a tighter reduction exists.

But, more importantly, the proof relies on the programmability of the random oracle. Considering the counterintuitive nature of programmability once the random oracle needs to be instantiated, this seems to be rather disconcerting.

Paillier and Vergnaud [15] analyzed the security of Schnorr Signatures in the standard model and showed that, if the one-more discrete logarithm assumption holds, the security of Schnorr signatures can be reduced neither to the discrete logarithm nor the one-more discrete logarithm problem – at least using algebraic reductions. Algebraic reductions are reductions that are limited to only apply group operations on group elements. They were first defined by Boneh and Venkatesan [6]. Paillier and Vergnaud use a – supposedly more natural – definition of algebraicity that 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.

In this paper we analyze to what extent the programmability of the random oracle is required to achieve provable security of the Schnorr signature scheme.

2 Preliminaries

2.1 Digital Signature Schemes

We recall the syntax of digital signature schemes in Definition 1.

Definition 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* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) \stackrel{?}{=} 1$.

We recall two common security notions for digital signature schemes. Namely the notions of existential unforgeability under adaptive chosen-message attacks in Definition 2 and strong existential unforgeability under adaptive chosen-message attacks in Definition 3.

Definition 2 (Existential Unforgeability)**.** *A signature scheme* $\mathcal{S}$ *is said to be* existentially unforgeable under adaptive chosen-message attacks (EUF-CMA) *if and only if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *the following success function is negligible in* κ*.*

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

where $\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa)$ *is the* existential unforgeability experiment *from Figure 2.1.*

Definition 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 2.1.*

$\mathsf{Exp}^{\mathcal{S},\mathcal{A}}_{\mathsf{EUF\text{-}CMA}}(\kappa)$:
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{S}.\mathsf{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk})$
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 2.1: (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 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 4.

Definition 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 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:*

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

2.2 Problems

Cryptographers use reductionist proofs to show that a primitive or scheme is secure. Such a proof works by constructing an efficient algorithm, called reduction, which – given access to an adversary against the scheme – is able to solve some computational or decisional problem. Under the assumption that the problem is hard, the existence of the reduction thus implies that the adversary cannot exist.

A problem can be seen 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 to a certain degree by providing access to some oracle. 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. For example for a decisional problem there always exists an adversary that simply guesses and wins with non-negligible probability. However, it may still be hard to do any better than this trivial algorithm. We therefore term this trivial algorithm *threshold algorithm* and an adversary is required to be better than this threshold to be considered successful.

To enable us to precisely argue about problems and reductions, we formally define a problem in Definition 6. Some important classes of problems are defined in Definition 7.

Definition 6 (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* **Inst**.

The computationally unbounded 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 $z \in$* **Inst** *and a candidate solution $x \in$* **Sol**. *The algorithm outputs $b \in \{0,1\}$. We say x is a valid solution to instance z if and only if $b \stackrel{?}{=} 1$.*

The efficient – and usually trivial – threshold algorithm Thresh *takes as input a problem instance z and outputs some $x \in$* **Sol**. *The threshold algorithm is a special adversary and as such also has access to* Orcl.

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

For a problem Π and an adversary $\mathcal{A}$ we define the following experiment:

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

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:*

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

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

Definition 7 (Some Specific Classes of Problems). *Let Π be a problem as defined in Definition 6.*

The problem Π is said to be non-interactive *if and only if $\Pi.\mathsf{Orcl}$ is the algorithm that always outputs $\perp$ and there is no 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 $\Pi.$***Sol** *is recursively enumerable, and the following holds:*

$$\forall z \leftarrow \Pi.\mathsf{IGen}(1^\kappa) : \left(\exists x \in \Pi.\textbf{Sol} : \Pi.\mathsf{Vrfy}(z, x) \stackrel{?}{=} 1\right).$$

The problem Π is said to be monotone *if and only if for all instances $z \leftarrow \Pi.\mathsf{IGen}(1^\kappa)$, all solutions $x \in \Pi.$***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 it 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.

2.2.1 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 8.

Definition 8 (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.*

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 is the interactive, efficiently generatable, monotone, and solvable *one-more discrete logarithm* problem first introduced by Bellare et al. [4]. It is 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 said oracle, hence, the name *one-more*. The problem with its corresponding hardness assumptions is formally described in Definition 9.

Definition 9 (One-More Discrete Logarithm Assumption [4])**.** *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 one-more discrete logarithm *($\mathsf{DL}_{\mathsf{OM}}$) assumption is said to hold over $\mathbb{G}$, if and only if for all positive integers n polynomial in κ 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) [19, 20] as derived from the Schnorr identification scheme via the Fiat-Shamir transform [9]. 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 10 (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* $\sigma := (R, c, y)$.

Signature Verification *The verification algorithm* $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$ *proceeds as follows: Parse* σ *as* (R, 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 [17, 18] still holds for randomly programming reductions. Randomly programming reductions as defined in [10] do not simulate the random oracle themselves. Instead, they work relative to a so called randomly programming random oracle. Whereas a conventional random oracle has only a single interface implementing a random mapping from **Dom** to **Rng**, a randomly programming random oracle has three interfaces, which allow for programming but only in a restricted sense.

Definition 11 (Randomly Programming Random Oracle)**.** *A* randomly programming random oracle *(RPRO) exposes three interfaces to the caller:*

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

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

Programming: *The programming interface* RO_{prog} *takes as input a pair* $(a, b) \in$ **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 1 (EUF-CMA Security of SSS Under Randomly Programming Reductions)**.** *The* discrete logarithm *problem over $\mathbb{G}$ is reducible to the* EUF-CMA *security of* SSS *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 [18]. What we need to show is that the limited programmability still allows the reduction to simulate the signing oracle and to apply the forking of the random oracle.

Simulating the Signing Oracle.

When asked to sign a message m, the reduction $\mathcal{R}$ chooses $b \xleftarrow{\$} \{0,1\}^\kappa$ and $y \xleftarrow{\$} \mathbb{Z}_q$, 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 (R, c, y).

Forking the Random Oracle.

To fork the random oracle 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 $b_i \xleftarrow{\$} \{0,1\}^\kappa$ and reprograms the oracle by calling $\mathsf{RO}_{prog}(a_i, b_i)$ before the original query is evaluated. Note that the distribution of $\mathsf{RO}_{eval}(a_i)$ is slightly biased, because of the possibility that b_is collide. However, as the size of the domain from which b_is are uniformly chosen is exponential in the security parameter, the collision probability – and therefore the bias – is obviously negligible. □

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.

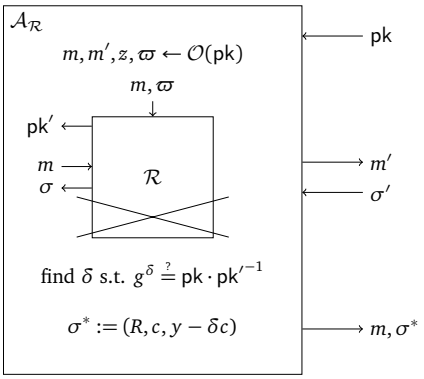

Figure 3.1: For each reduction $\mathcal{R}$, the associated inefficient adversary $\mathcal{A}_{\mathcal{R}}$ works by choosing messages, a group element, and a random tape according to a random function. It proceeds to forge a signature by internally simulating an instance of $\mathcal{R}$ on the group 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.

3.2 Schnorr Signatures are not provably secure under simple non-programming reductions

We now show that the Schnorr Signature Scheme cannot be proven existentially unforgeable 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 one-more discrete logarithm assumption does not hold over $\mathbb{G}$.

We term the restricted class of reductions as *simple reductions*. Such simple reductions only ever 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 the original security reduction by Pointcheval and Stern [18] is of this type.

As the reductions are not allowed to program the random oracle. We thus call them non-programming reductions. Instead of simulating the random oracle itself, thus allowing it to program, a non-programming reduction works relative to an external fixed random function and it is required to honestly answer all random oracle queries.

Theorem 2 (Non-Programming Irreducibility for SSS)**.** *Assume that the* one-more discrete logarithm *assumption holds over* $\mathbb{G}$*. Then, there exists no simple non-programming reduction that reduces the* EUF-CMA *security of* SSS *over* $\mathbb{G}$ *to the discrete logarithm problem over* $\mathbb{G}$*.*

More precisely, assume there exists a simple non-programming black-box reduction $\mathcal{R}$ *that converts any adversary* $\mathcal{A}$ *against the* EUF-CMA *security of* SSS *working in group* $\mathbb{G}$ *into an adversary against the* DL *problem over* $\mathbb{G}$*.*[1] *Further assume the reduction has success probability* $\mathsf{Succ}^{\mathcal{R}^{\mathcal{A}}}_{\mathsf{DL},\mathbb{G}}(\kappa)$ *and runtime* $\mathsf{Time}_{\mathcal{R}}(\kappa)$*. Then, there also exists a successful (but possibly inefficient) adversary* $\mathcal{A}_{\mathcal{R}}$ *against the* EUF-CMA *security of* SSS *and a meta-reduction* $\mathcal{M}$ *that is an adversary against* n-DL *over* $\mathbb{G}$ *with success probability* $\mathsf{Succ}^{\mathcal{M}}_{n\text{-}\mathsf{DL},\mathbb{G}}(\kappa) = \left(\mathsf{Succ}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R}}}}_{\mathsf{DL},\mathbb{G}}(\kappa)\right)^2$ *and runtime* $\mathsf{Time}_{\mathcal{M}}(\kappa) = 2 \cdot \mathsf{Time}_{\mathcal{R}}(\kappa) + \mathsf{poly}(\kappa)$.

Proof. Roughly speaking, the meta-reduction $\mathcal{M}$ works as follows: It invokes two instances of $\mathcal{R}$, on the inputs z_0 and z_1, respectively, and independent random tapes. When the instances invoke the forger with public key pk_0 and pk_1 respectively, $\mathcal{M}$ simulates a specific forger, we will describe first. 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 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. For the discrete logarithms of the inputs z_2 through z_n $\mathcal{M}$ simply queries the $\mathsf{DL}_{\mathsf{OM}}$ oracle. As we are working in the (non-programmable) random oracle model, an outside reduction expects to see all the random oracle queries, the simulated adversary would issue. The meta-reduction $\mathcal{M}$ therefore makes sure issue exactly those queries.

The Adversary $\mathcal{A}_{\mathcal{R}}$.

For every reduction $\mathcal{R}$, we construct an (inefficient) adversary $\mathcal{A}_{\mathcal{R}}$ – depicted in Figure 3.1. Initially, $\mathcal{A}_{\mathcal{R}}$ chooses a random function $\mathcal{O} : \mathbb{G} \to \{0,1\}^{\kappa} \times \{0,1\}^{\kappa} \times \mathbb{G} \times \{0,1\}^{\mathsf{poly}(\kappa)}$. It then receives as input a public key pk and computes two messages $m, m' \in \{0,1\}^{\kappa}$, a group element $z \in \mathbb{G}$, and a random tape ϖ of suitable length for $\mathcal{R}$ by evaluating the

[1] 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 $\mathsf{pk} \in \mathbb{G}$.

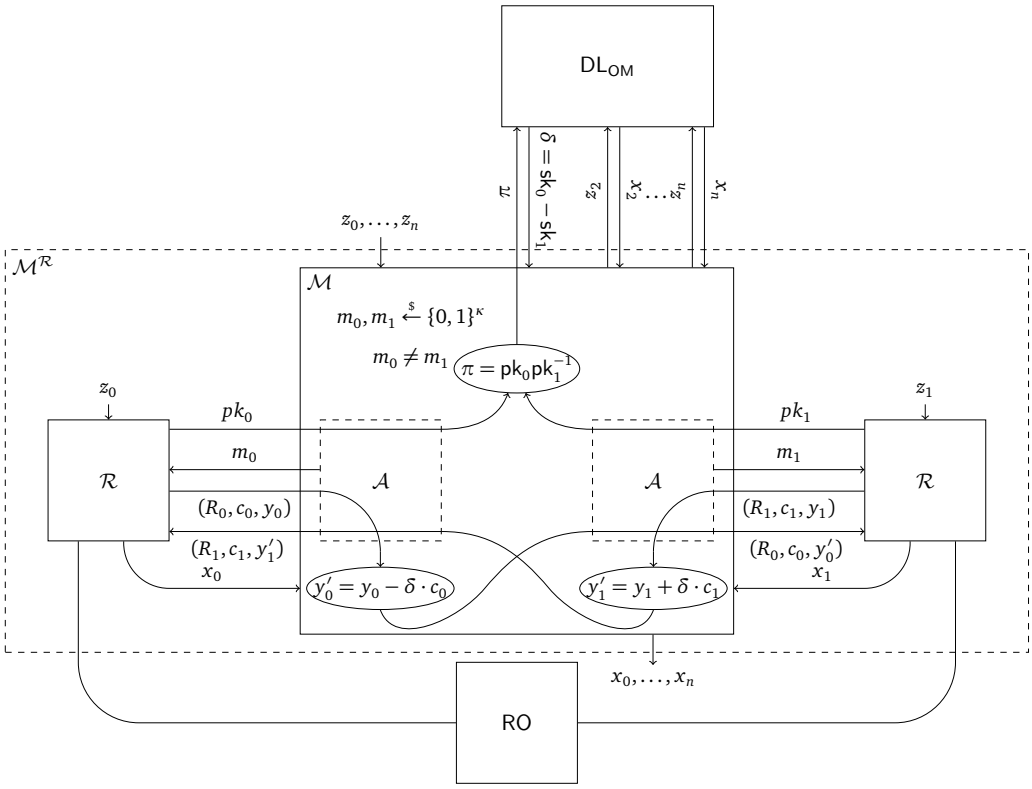

Figure 3.2: The meta-reduction uses two instances of $\mathcal{R}$ and simulates the adversary $\mathcal{A}$ by obtaining the difference between the secret keys and adapting the signatures output by $\mathcal{R}$ to the other key, respectively.

random function $(m, m', z, \varpi) \leftarrow \mathcal{O}(\mathsf{pk})$. It then queries m' to its signing oracle. If the resulting signature is invalid, $\mathcal{A}_{\mathcal{R}}$ aborts. Otherwise, the signature is discarded and $\mathcal{A}_{\mathcal{R}}$ invokes an internal copy of $\mathcal{R}$ – denoted $\mathcal{R}^*$ in the following – on input z and random tape ϖ. Any queries made by $\mathcal{R}^*$ to the random oracle are simply forwarded by the adversary to its own random oracle. When $\mathcal{R}^*$ invokes the forger by outputting a public key pk', $\mathcal{A}_{\mathcal{R}}$ queries m to $\mathcal{R}^*$. If $\mathcal{R}^*$ is unable to sign m, $\mathcal{A}_{\mathcal{R}}$ aborts. Otherwise it obtains a signature $\sigma = (R, c, y)$ on m, aborts $\mathcal{R}^*$, and proceeds to forge a signature. Using its unbounded computational power, $\mathcal{A}_{\mathcal{R}}$ finds δ such that $\mathsf{pk} \cdot \mathsf{pk}'^{-1} = g^{\delta}$, i.e., $\delta = \mathsf{sk} - \mathsf{sk}'$. Given this difference of the secret keys, it then adapts σ to the public key pk by computing $\sigma^* = (R^* := R, c^* := c, y^* := y - \delta \cdot c)$. Finally, it returns the forgery (m, σ^*). Note that $\mathcal{A}_{\mathcal{R}}$ will always return the same message-signature pair, even if it is rewound, as all values used to compute the pair are uniquely determined by pk and $\mathcal{O}$. Further, observe that for any reduction $\mathcal{R}$, $\mathcal{A}_{\mathcal{R}}$ is successful with the same probability with which an instance of $\mathcal{R}$ is able to sign a randomly chosen messages. This probability must obviously be non-negligible and, therefore, $\mathcal{A}_{\mathcal{R}}$ is a successful adversary against the EUF-CMA security of SSS. As the reduction $\mathcal{R}$ must work with any EUF-CMA adversary against SSS, it follows directly that $\mathcal{R}^{\mathcal{A}_{\mathcal{R}}}$ must have non-negligible success probability and run in polynomial time.

As stated in the informal description of the meta-reduction above, it is necessary to consider which random oracle queries the $\mathcal{A}_{\mathcal{R}}$ would issue. These queries are exactly those issued by $\mathcal{R}^*$ up to the point where it answers the first signature query. These queries are all issued *after* an outside reduction has already answered the signing query. 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 3.2 – on input $z_0, \ldots, z_n \in \mathbb{G}$ 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 the two instances of the reduction are answered by simply 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 $\mathcal{A}$, $\mathcal{M}$ now proceeds as follows:

1. Obtain $\mathsf{pk}_0 \leftarrow \mathcal{R}_0$, $\mathsf{pk}_1 \leftarrow \mathcal{R}_1$.
2. Choose $m_0, m_1 \xleftarrow{\$} \{0,1\}^{\kappa}$ with $m_0 \neq m_1$.
3. Query $(R_0, c_0, y_0) \leftarrow \mathcal{R}_0.\mathsf{Sign}(m_0)$, $(R_1, c_1, y_1) \leftarrow \mathcal{R}_1.\mathsf{Sign}(m_1)$.

4. If either $\mathcal{R}_0$ or $\mathcal{R}_1$ were unable to provide a valid signature, abort.
5. Let $\mathbb{Q}_{\mathcal{H},i}$ be the sequence of random oracle queries issued by $\mathcal{R}_i$ up to, and including, step 3.
6. 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 $\mathcal{R}_1$.
7. Query $\delta \leftarrow \mathsf{DL}_{\mathsf{OM}}(\mathsf{pk}_0\mathsf{pk}_1^{-1})$.
8. Adapt signatures: $\sigma_0' := (R_0, c_0, y_0 - \delta \cdot c_0)$, $\sigma_1' := (R_1, c_1, y_1 + \delta \cdot c_1)$.
9. Return (m_1, σ_1') as a forgery to $\mathcal{R}_0$ and (m_0, σ_0') as a forgery to $\mathcal{R}_1$.

It is possible that $\mathcal{M}$ is not able to obtain a signature from one of the reduction instances in step 3, because the reduction already output a solution. This may be the case, if the instance either never invokes the adversary or chooses not to answer the sign query and directly outputs the solution instead. This is, however, not a problem. In this case, 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 all remaining inputs to the $\mathsf{DL}_{\mathsf{OM}}$ oracle and output the solutions. In the following we can therefore assume, that the reductions actually answer the signature queries.

If $\mathcal{R}_b$ 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 $\mathcal{A}_{\mathcal{R}}$. 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. For $i \in \{2, \ldots, n\}$, $\mathcal{M}$ then queries $x_i \leftarrow \mathsf{DL}_{\mathsf{OM}}(z_i)$ and finally outputs $x_0, \ldots, x_n$.

Observe that $\mathcal{M}$ uses the $\mathsf{DL}_{\mathsf{OM}}$ oracle to obtain $n-1$ discrete logarithms directly and only once to simulate both forgers. Thus, it is easy to see that, whenever $\mathcal{M}$ does not abort, it makes exactly n queries to the $\mathsf{DL}_{\mathsf{OM}}$ oracle. Therefore, $\mathcal{M}$ is a valid adversary in the n-DL experiment. This is also the point where it is crucial that $\mathcal{R}$ is *simple*, confer Remark 1.

The runtime of $\mathcal{M}$ consists of 2 executions of $\mathcal{R}$, n oracle calls, and a constant number of modular inversions, multiplications, and additions. Therefore, it holds that $\mathsf{Time}_{\mathcal{M}}(\kappa) = 2 \cdot \mathsf{Time}_{\mathcal{R}}(\kappa) + \mathsf{poly}(n)$.

Further, 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, whether 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:

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

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 cannot be the case, because $\mathcal{M}$ perfectly simulates $\mathcal{A}_{\mathcal{R}}$. Both algorithms compute the forgery as follows: Run an instance of $\mathcal{R}$ on random input and an independent random tape. Obtain a signature on a 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 of a public key, first query a random message to the signing oracle. They then issue exactly those random oracle queries, an instance of $\mathcal{R}$ issues up until it 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, $\mathsf{Succ}_{n\text{-}\mathsf{DL}}^{\mathcal{M}}(\kappa) = (\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{M}}}(\kappa))^2 = (\mathsf{Succ}_{\mathsf{DL},\mathbb{G}}^{\mathcal{R}^{\mathcal{A}_{\mathcal{R}}}}(\kappa))^2$, which is non-negligible by assumption. □

Remark 1. *Note that the restriction to simple 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 $n+1$ queries to the $\mathsf{DL}_{\mathsf{OM}}$ oracle and could, thus, no longer win in the n-DL experiment.*

Remark 2. *While it may seem strange at first that the adversary $\mathcal{A}_{\mathcal{R}}$ 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 adversary $\mathcal{A}_{\mathcal{R}}$.*

It should be noted, that the meta-reduction employed in the proof of Theorem 2 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, 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.

4 Limits of the Meta-Reduction Technique

Paillier and Vergnaud [15], as well as ourselves, have used meta-reductions to provide evidence that – once we drop programmability – the security of Schnorr signatures might not be equivalent to the discrete logarithm problem after all. Both results are not definitive proofs insofar, as they both are partial. Paillier and Vergnaud rule out algebraic reductions in the standard model. We rule out *simple* non-programming reductions. There could – at least theoretically – still exist a reduction that does not fall into those classes.

However, it is interesting to note that in both cases the meta-reduction-based proofs rely on the one-more discrete logarithm assumption. As the discrete logarithm assumption does not seem to imply its one-more variant [7], 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."

We actually show that for any randomized signature scheme $\mathcal{S}$ – such as any Fiat-Shamir transformed scheme – finding a meta-reduction to a non-interactive problem – such as the discrete logarithm 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.

4.1 An Inefficient Reduction for any Randomized Signature Scheme

Let $\mathcal{S}$ be a randomized signature scheme and let Π_1 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. Further, let p be the maximum number of signature queries issued by a forger $\mathcal{A}$ and assume that $\mathcal{R}$ knows p.

The reduction $\mathcal{R}$ – depicted in Figure 4.1 – on input an instance z of Π_1 first generates a key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\kappa)$, initializes the counter variable $i := 0$, and chooses a random function $\mathcal{O} : \{0,1\}^{2^\kappa} \times \mathbb{Z}_p \to \mathcal{C}oins_{\mathsf{pk}}$. The public key pk is then output as the key under which the forger is supposed to forge a signature.

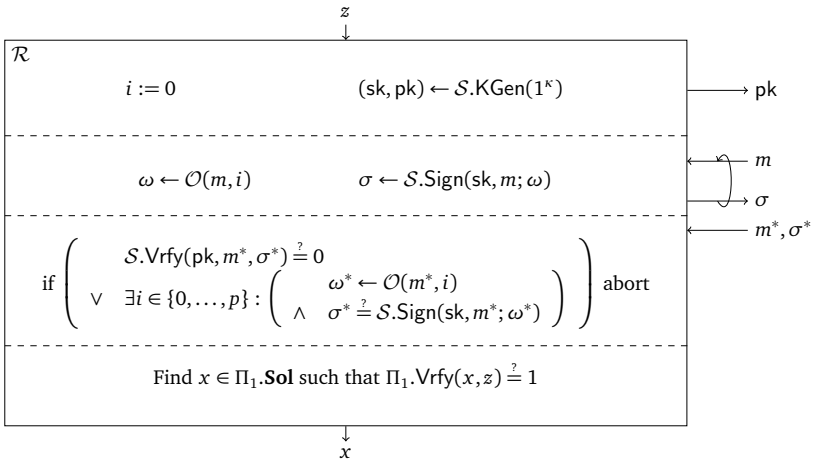

Figure 4.1: The reduction uses a random function $\mathcal{O}$ to decide whether it is being re-fed its own output. This way it can abort if it is used in such a way by a potential meta-reduction.

When the forger queries a message m to the signing oracle, $\mathcal{R}$ determines random coins $\omega \leftarrow \mathcal{O}(m, i)$, 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 $p+1$, $\mathcal{R}$ 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., $\mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m^*, \sigma^*) \stackrel{?}{=} 0$, $\mathcal{R}$ 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, $\mathcal{R}$ also immediately aborts. Otherwise, $\mathcal{R}$ enumerates all possible solutions $x \in \Pi_1.\mathbf{Sol}$ and checks whether $\Pi_1.\mathsf{Vrfy}(x, z) \stackrel{?}{=} 1$. Once such an x is found, it is output by $\mathcal{R}$ as the solution. Because Π_1 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 EUF-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

$$\mathsf{Succ}_{\Pi_1}^{\mathcal{R}^{\mathcal{A}}}(\kappa) \geq (1 - p \cdot \epsilon(\kappa)) \cdot \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) = \mathsf{Succ}_{\mathsf{EUF\text{-}CMA}}^{\mathcal{S},\mathcal{A}}(\kappa) - \epsilon'(\kappa)$$

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 Π_1 to the EUF-CMA security of $\mathcal{S}$.

Lemma 1 (Meta-Reductions Cannot Replay Signatures)**.** *Let $\mathcal{M}$ be a non-programming black-box meta-reduction that converts any* $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_1)$ *reduction into an adversary against some problem Π_2. Let $\mathbb{Q}$ be the set of message-signature pairs (m_i, σ_i) resulting from $\mathcal{M}$'s queries to the signing oracle, and let (m^*, σ^*) be the message-signature pair output by $\mathcal{M}$ as a forgery, while simulating 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 simple. 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 $\mathsf{Succ}_{\Pi_1}^{\mathcal{R}^{\mathcal{M}}}(\kappa) = 0$ for $\mathcal{M}$ if it replays an element of $\mathbb{Q}$ as 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 3 (Meta-Reductions to Non-Interactive Problems Are Hard)**.** *Let $\mathcal{S}$ be a randomized signature scheme, Π_1 be a monotone solvable problem, and Π_2 be a non-interactive, efficiently generatable problem. If Π_2 is hard, then finding an efficient meta-reduction $\mathcal{M}$ that converts any successful* $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_1)$*-reduction into an efficient successful adversary against Π_2 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 $(\mathsf{EUF\text{-}CMA}_{\mathcal{S}} \rightsquigarrow \Pi_1)$*-reduction into an adversary against Π_2. 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 success probability* $\mathsf{Succ}_{\mathsf{sEUF\text{-}CMA}}^{\mathcal{S},\mathcal{N}^{\mathcal{M}}}(\kappa) \geq \frac{1}{r} \cdot \mathsf{Succ}_{\Pi_2}^{\mathcal{M}^{\mathcal{R}}}(\kappa)$ *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 number of reduction instances invoked by $\mathcal{M}$.*

Proof. We construct $\mathcal{N}$ – depicted in Figure 4.2 – such that, from $\mathcal{M}$'s perspective, it is perfectly indistinguishable from the reduction $\mathcal{R}$. On input a public key pk of $\mathcal{S}$ the meta-meta-reduction $\mathcal{N}$ chooses a random $i \stackrel{\$}{\leftarrow} \mathbb{Z}_r$ and sets $\mathsf{pk}_i = \mathsf{pk}$. For all $j \in \mathbb{Z}_r \setminus \{i\}$, $\mathcal{N}$ computes $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathsf{KGen}(1^\kappa)$. Then it generates a random instance $x \leftarrow \Pi_2.\mathsf{IGen}(1^\kappa)$ and invokes $\mathcal{M}$ on x.

When $\mathcal{M}$ invokes the jth instance of $\mathcal{R}$ on an instance x_j of Π_1, $\mathcal{N}$ 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 $\mathcal{M}$. 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 will be completely independent. 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 necessary 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

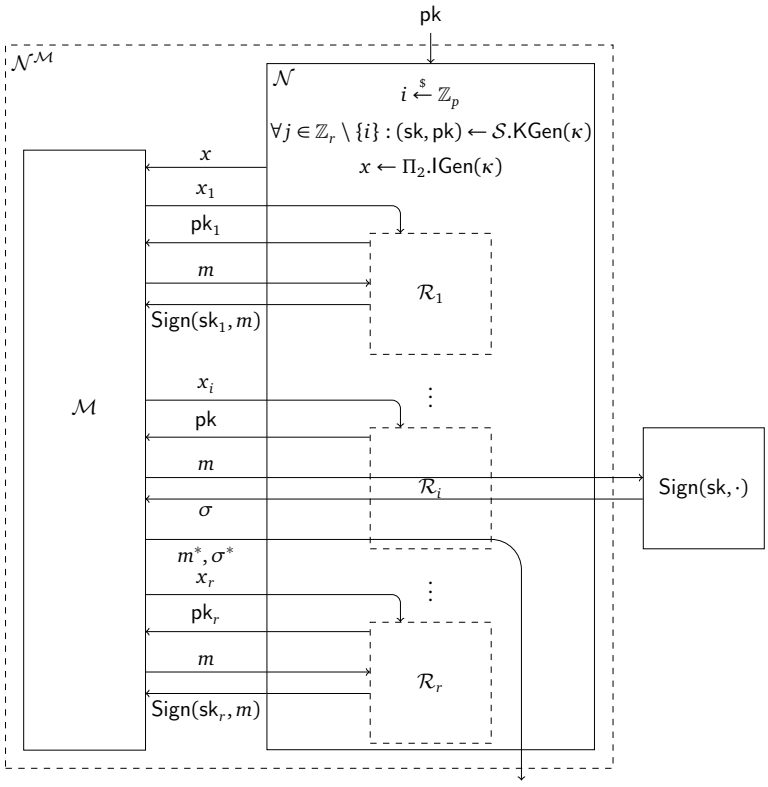

Figure 4.2: The meta-meta-reduction relies on the fact that $\mathcal{M}$ cannot replay an old signature. It guesses the reduction instance, for which $\mathcal{M}$ will output a forgery, embeds the public key in that instance, and outputs the forgery provided by $\mathcal{M}$.

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 p instances of $\mathcal{R}$. The public key in the sEUF-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 true 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 $\mathcal{S}$.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 p 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 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 sEUF-CMA game. As i is chosen uniformly and $\mathcal{M}$ is perfectly oblivious about i, we get $\mathsf{Succ}^{\mathcal{S},\mathcal{N}^{\mathcal{M}}}_{\text{sEUF-CMA}}(\kappa) \geq \frac{1}{r} \cdot \mathsf{Succ}^{\mathcal{M}^{\mathcal{R}}}_{\Pi_2}(\kappa)$ as claimed. Besides executing $\mathcal{M}$ the meta-meta-reduction $\mathcal{N}$ needs to generate an instance of Π_2, generate $r - 1$ key pairs, and answer a polynomial number of sign queries. All of this is possible in polynomial time, because Π_2 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. □

5 Conclusion and Open Problems

We have shown that it is unlikely that the security of the Schnorr signature scheme could be reduced to the discrete logarithm problem without programming the random oracle. As with the results by Paillier and Vergnaud [15], our meta-reduction needs to rely on the one-more variant of the discrete logarithm problem and is, thus, not an unconditional result on the security of Schnorr signatures. However, we also show that a stronger result is unlikely to be achieved. In fact, we proof that for any randomized signature scheme, finding a meta-reduction from a non-interactive problem is just as hard as finding an adversary against the scheme's strong unforgeability. While this is no impossibility result, consider the following: If the scheme is not obviously rerandomizable, finding an adversary against strong unforgeability is most likely no easier than finding one against normal unforgeability. If it was easy to find an adversary, there would be no need for a meta-reduction, because the insecurity would already be settled.

The question, whether the proof could be strengthened in such a way that the meta-meta-reduction also breaks the EUF-CMA security, remains open.

Bibliography

[1] M. Abdalla, J. An, M. Bellare, and C. Namprempre, *From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security*, in Advances in Cryptology — EUROCRYPT 2002, vol. 2332 of LNCS, Springer, 2002, pp. 418–433.

[2] M. Abe, J. Groth, and M. Ohkubo, *Separating short structure-preserving signatures from non-interactive assumptions*, in Advances in Cryptology — ASIACRYPT 2011, vol. 7073 of LNCS, Springer, 2011, pp. 628–646.

[3] F. Baldimtsi and A. Lysyanskaya, *On the security of one-witness blind signature schemes*. Cryptology ePrint Archive, Report 2012/197, 2012. http://eprint.iacr.org/2012/197.

[4] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, *The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme*, Journal of Cryptology, 16 (2003), pp. 185–215.

[5] M. Bellare and P. Rogaway, *Random oracles are practical: A paradigm for designing efficient protocols*, in ACM Conference on Computer and Communications Security (CCS '93), ACM, 1993, pp. 62–73.

[6] D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring*, in Advances in Cryptology — EUROCRYPT 1998, vol. 1403 of LNCS, Springer, 1998, pp. 59–71.

[7] E. Bresson, J. Monnerat, and D. Vergnaud, *Separation results on the "One-More" computational problems*, in Topics in Cryptology — CT-RSA 2008, vol. 4964 of LNCS, Springer, 2008, pp. 71–87.

[8] J.-S. Coron, *Optimal security proofs for PSS and other signature schemes*, in Advances in Cryptology — EUROCRYPT 2002, vol. 2332 of LNCS, Springer, 2002, pp. 272–287.

[9] A. Fiat and A. Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, in Advances in Cryptology — CRYPTO 1986, vol. 263 of LNCS, Springer, 1986, pp. 186–194.

[10] M. Fischlin, A. Lehmann, T. Ristenpart, T. Shrimpton, M. Stam, and S. Tessaro, *Random oracles with(out) programmability*, in Advances in Cryptology — ASIACRYPT 2010, vol. 6477 of LNCS, Springer, 2010, pp. 303–320.

[11] M. Fischlin and D. Schröder, *On the impossibility of three-move blind signature schemes*, in Advances in Cryptology — EUROCRYPT 2010, vol. 6110 of LNCS, Springer, 2010, pp. 197–215.

[12] C. Gentry and D. Wichs, *Separating succinct non-interactive arguments from all falsifiable assumptions*, in 43rd ACM Symposium on Theory of Computing (STOC 2011), ACM, 2011, pp. 99–108.

[13] S. Goldwasser and Y. Tauman Kalai, *On the (in)security of the Fiat-Shamir paradigm*, in 44th Symposium on Foundations of Computer Science (FOCS 2003), IEEE Computer Society, 2003, pp. 102–113.

[14] J. B. Nielsen, *Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case*, in Advances in Cryptology — CRYPTO 2002, vol. 2442 of LNCS, Springer, 2002, pp. 191–214.

[15] P. Paillier and D. Vergnaud, *Discrete-log-based signatures may not be equivalent to discrete log*, in Advances in Cryptology — ASIACRYPT 2005, vol. 3788 of LNCS, Springer, 2005, pp. 1–20.

[16] R. Pass, *Limits of provable security from standard assumptions*, in 43rd ACM Symposium on Theory of Computing (STOC 2011), ACM, 2011, pp. 109–118.

[17] D. Pointcheval and J. Stern, *Security proofs for signature schemes*, in Advances in Cryptology — EUROCRYPT 1996, vol. 1070 of LNCS, Springer, 1996, pp. 387–398.

[18] D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures*, Journal of Cryptology, 13 (2000), pp. 361–396.

[19] C.-P. Schnorr, *Efficient identification and signatures for smart cards*, in Advances in Cryptology — CRYPTO 1989, vol. 435 of LNCS, Springer, 1990, pp. 239–252.

[20] ———, *Efficient signature generation by smart cards*, Journal of Cryptology, 4 (1991), pp. 161–174.

[21] Y. Seurin, *On the exact security of Schnorr-type signatures in the random oracle model*, in Advances in Cryptology — EUROCRYPT 2012, vol. 7237 of LNCS, Springer, 2012, pp. 554–571.