Discrete-Log-Based Signatures May Not Be Equivalent to Discrete Log

Pascal Paillier[1] and Damien Vergnaud[2]

[1] Gemplus Card International, Advanced Cryptographic Services
34, rue Guynemer, 92447 Issy-les-Moulineaux Cedex, France
`pascal.paillier@gemplus.com`
[2] Laboratoire de Mathématiques Nicolas Oresme
Université de Caen, Campus II, B.P. 5186,
14032 Caen Cedex, France,
`vergnaud@math.unicaen.fr`

Abstract. We provide evidence that the unforgeability of several discrete-log based signatures like Schnorr signatures cannot be equivalent to the discrete log problem in the standard model. This contradicts in nature well-known proofs standing in weakened proof methodologies, in particular proofs employing various formulations of the Forking Lemma in the random oracle Model. Our impossibility proofs apply to many discrete-log-based signatures like ElGamal signatures and their extensions, DSA, ECDSA and KCDSA as well as standard generalizations of these, and even RSA-based signatures like GQ. We stress that our work sheds more light on the provable (in)security of popular signature schemes but does not explicitly lead to actual attacks on these.

1 Introduction

It is striking to observe that after more that two decades of active research on the matter, the standard-model security of discrete-log based signatures like Schnorr, ElGamal or DSA remains mysteriously unknown. Although dedicated proof techniques do exist in weakened models (*e.g.* the random oracle model (ROM) [19,4,8] or the generic group model (GGM) [7]), none of them provides intuition about the actual security of discrete-log signatures. Even though they have withstood concerted cryptanalytic effort fairly well, we suspect that the real-life security of many of these signature schemes is actually weaker than expected. We provide evidence that most discrete-log-based signatures defined over some prime-order group $\mathbb{G}$ cannot be equivalent to extracting discrete logs over $\mathbb{G}$ in the standard model. Our results are partial in the sense that we disprove equivalence via *algebraic* reductions. In brief, algebraic reductions can only apply group operations on group elements. This restriction is not overly restrictive as we do not know any example of a cryptographic reduction which is not algebraic. Our results suggest that most discrete-log based signature schemes just cannot reach a maximal security level *i.e.* equivalence towards their primitive problem, or that if some of them do, it is through non-algebraic reductions exploiting intricate and subtle relations within the group $\mathbb{G}$.

Most interestingly, our work highlights a possible separation between the standard model and the random oracle model in which it is well-known that forging Schnorr signatures (for instance) is equivalent to extracting discrete logs. An interpretation is that random-oracle-based proofs leave unfair advantage to security reductions by probing and modifying the adversary's internal computations and thereby letting the random oracle play a crucial role that cannot be justified in real life. Previous works have observed similar separations in specific contexts [2,18].

The Fiat-Shamir paradigm of transforming identification schemes into digital signature schemes [13] is popular because it yields efficient protocols. However all known results for the security of Fiat-Shamir-transformed signature schemes like Schnorr take place in the ROM[1]. Even worse, they impose the loss of a factor nearly q_H (the number of queries the forger makes to the random oracle) in either execution time or success probability of reductions that convert a forger into an algorithm that extracts discrete logarithms. While no proof exists that the loss of this factor is necessary, the problem seems inherent to the way signature schemes are constructed from identification protocols.

We prove in this paper that any random-oracle-based reduction from computing discrete logarithms to forging Schnorr signatures *must* lose a factor at least $\sqrt{q_H}$. This shows that a proof of equivalence in the ROM, if algebraic, *will never be tight.* We believe our work gives a new perspective as to why no efficient proof of equivalence to the discrete log problem has ever been found for Schnorr signatures despite considerable research efforts.

We emphasize that although our work disproves that Schnorr, ElGamal, DSA, GQ, etc. are maximally secure, no actual attack or weakness of either of these signature schemes arises from our impossibility results. Nothing stated here refutes that forging signatures is likely to be intractable in practice.

1.1 Our contributions

Our results are manyfold. Introducing a simple way to simulate forgeries, we are able to relate security properties of many signature schemes (Schnorr, (Meta) ElGamal, DSA[2], ECDSA, KCDSA, GQ) to *one-more* computational problems, in a positive or *negative* sense. In the positive sense, we prove the unbreakability of these signatures (meaning that the signing key cannot be recovered) under chosen-message attacks, thereby identifying security properties that have remained unknown for these schemes.

[1] It is known that the Fiat-Shamir transform provides a separation between the ROM and the standard model, see [14].

[2] Note that this work constitutes the first proper security analysis of DSA and ECDSA in the standard model. Previous to this work the only known security result on DSA schemes was that of Brown on ECDSA which assumed a generic group [7].

Starting from the same simulation technique, we show that no algebraic reduction can exist that would relate the unforgeability (under any kind of attacks) of these signatures to their primitive problem. This result is extendable to the one-more setting, meaning that there cannot exist a similar reduction to a weakened, one-more version of the primitive problem. Our impossibility proofs rely on the construction of an efficient meta-reduction relating such a reduction to the one-more problem itself. Thus, under the assumption that this problem is intractable, the fact that a polynomial meta-reduction exists forbids the existence of algebraic reductions. We note that our meta-reductions are *perfect* meaning that they preserve success probabilities perfectly. This emphasizes the strength of our impossibility results.

1.2 Roadmap

We start by providing definitional facts about discrete-log-based signature schemes, security notions for signatures, the discrete log and one-more discrete log assumptions over a group $\mathbb{G}$, reductionist security proofs and algebraic reductions. Section 3 proves that Schnorr signatures are unbreakable under a chosen-message attack. Section 4 then proves that if the one-more discrete log assumption holds, then Schnorr signatures cannot be proven equivalent to the discrete log problem and Section 5 further extends this impossibility to the one-more discrete log problem. Section 6 then applies our proof technique to other signatures schemes, slightly adapting the proof to the underlying computational problem when necessary. Lastly, Section 7 explores the case of random-oracle-based reductions and shows that any reduction of that type, if algebraic, must loose a factor close to $\sqrt{q_H}$. We conclude with a series of open questions in Section 8.

2 Preliminaries

2.1 Schnorr Signatures

Schnorr's identification protocol was introduced in the late eighties [21,20] as a means to prove knowledge of the discrete logarithm of a publicly known group element. Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q and P and V denote a prover and a verifier. By engaging in the protocol, P proves to V that he knows the discrete log x of a public group element $y = g^x$. The protocol has three simple moves. (Commitment) P selects a random $k \overset{\$}{\leftarrow} \mathbb{Z}_q$, computes $r = g^k$ and sends r to V. (Challenge) V picks a random $c \overset{\$}{\leftarrow} \mathbb{Z}_q$ and sends c to P. (Response) P computes and sends $s = k + cx \bmod q$ to V. Lastly, V verifies that $g^s \cdot y^{-c} = r$ and recognizes that P knows x if the equality holds.

Schnorr signatures derive from Schnorr's identification protocol by applying the Fiat-Shamir transform [13] with respect to a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$. The Fiat-Shamir-transformed protocol is changed into a signature scheme by making it non-interactive. In this respect, the signer acts like P and simulates a verifier V by computing the challenge c himself as $c = H(m, r)$. For concreteness, we detail Schnorr's signature scheme Σ_H as a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.

Key generation. Gen selects a random $x \xleftarrow{\$} \mathbb{Z}_q$. The secret key is x while the public key is $y = g^x \in \mathbb{G}$.

Signing procedure. Given a message $m \in \{0,1\}^\star$, $\textsc{Sign}(m)$ picks a random $k \xleftarrow{\$} \mathbb{Z}_q$, computes $r = g^k$, $c = H(m, r)$ and $s = k + cx \bmod q$. The output signature is (s, c).

Verification procedure. $\textsc{Ver}(m, (s, c))$ returns 1 if $H(m, g^s y^{-c}) = c$ and 0 otherwise.

Schnorr signatures constitute one of the most important ingredients in the design of cryptographic protocols, cryptosystems and proofs of knowledge.

2.2 Security Notions

Security notions for signature schemes are defined with respect to several types of adversaries or equivalently, as the conjunction of an adversarial goal and an attack scenario. An adversary is modeled as a probabilistic Turing machine attempting to fulfill the goal while given access to certain resources when interacting with the signature scheme.

Adversarial goals. We make use of three separate goals in this paper although others may also be of interest (*e.g.* signature *malleability* [19]). We say that a signature scheme is *breakable* (BK) when an adversary extracts the secret key matching a prescribed public key. The scheme is said to be *universally forgeable* (UF) when there exists an adversary $\mathcal{A}$ that returns a valid signature on a message given as input to $\mathcal{A}$. The notion of *existential forgeability* (EF) is similar but allows the adversary to choose freely the value of the signed message.

Attack models. We consider two attack scenarios in this paper. In a *key-only attack* (KOA), the adversary is given nothing else than a public key as input[3]. In a *chosen-message attack* (CMA), the adversary is given adaptive access to signatures on messages of his choice while attempting to achieve his goal.

Security notions are obtained by coupling an adversarial goal with an attack model. We distinguish between several notions of reference for which general results are immediate, as shown on Figure 1. We refer the reader to the extensive cryptographic literature for a more formal definition of these security notions.

2.3 Discrete Logarithm Problems

DL. Solving the discrete log problem $\mathsf{DL}[g, r]$ in a group $\mathbb{G} = \langle g \rangle$ of prime order q consists in computing $k \in \mathbb{Z}_q$ given $r = g^k \in \mathbb{G}$. Because of its random self-reducibility [19], the hardness of the discrete log problem is essentially independent from the choice of its inputs (g, r) and rather depends on the inner

[3] The term *no-message attacks* is also frequently used to designate such attacks.

Existential forgeries	$\mathsf{EF\text{-}KOA}\,[\mathcal{S}]$	$\Rightarrow$	$\mathsf{EF\text{-}CMA}\,[\mathcal{S}]$
	$\Uparrow$		$\Uparrow$
Universal forgeries	$\mathsf{UF\text{-}KOA}\,[\mathcal{S}]$	$\Rightarrow$	$\mathsf{UF\text{-}CMA}\,[\mathcal{S}]$
	$\Uparrow$		$\Uparrow$
Breakability	$\mathsf{BK\text{-}KOA}\,[\mathcal{S}]$	$\Rightarrow$	$\mathsf{BK\text{-}CMA}\,[\mathcal{S}]$
Goal vs. Attack	**Key only**		**Chosen message**

Fig. 1. Major security notions for signature schemes. $\mathcal{S}$ denotes an arbitrary signature scheme and $P_1 \Leftarrow P_2$ means that P_1 is polynomially reducible to P_2. Security notions are defined by their underlying problem *e.g.* $\mathsf{UF\text{-}KOA}\,[\mathcal{S}]$ denotes the problem of computing a universal forgery under a key-only attack.

structure of the group $\mathbb{G}$ itself. We denote DL the problem of computing discrete logs over $\mathbb{G} = \langle g \rangle$ with respect to a fixed base g. A probabilistic algorithm $\mathcal{A}$ that (ε, τ)-solves DL is such that

$$\Pr_{k \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}(g^k) = k\right] \geq \varepsilon$$

where the probability is taken over the random tape of $\mathcal{A}$ and $\mathcal{A}$ stops after time at most τ. The (ε, τ)-discrete-log assumption tells that DL cannot be (ε, τ)-solved over $\mathbb{G}$. The (asymptotic) discrete log assumption tells that if DL can be (ε, τ)-solved for $\tau = \mathsf{poly}\,(\log q)$ then ε is negligible before $1/\mathsf{poly}\,(\log q)$.

The One-More DL. The computational problem n-DL is defined as a natural extension of DL. A probabilistic algorithm $\mathcal{A}$ solving n-DL is given $n + 1$ group elements $r_0, r_1, \ldots, r_n$ as well as a limited access to a discrete log oracle $\mathsf{DL}_{\mathsf{OM}}$. $\mathcal{A}$ is allowed to access $\mathsf{DL}_{\mathsf{OM}}$ at most n times, thus obtaining the discrete logarithm of n group elements of his choice with respect to a fixed base g. $\mathcal{A}$ must eventually output the $n + 1$ discrete logs $k_0 = \mathsf{dl}_g\,(r_0)\,, \ldots, k_n = \mathsf{dl}_g\,(r_n)$. An algorithm $\mathcal{A}$ is said to (ε, τ)-solve n-DL when

$$\Pr_{k_0, \ldots, k_n \xleftarrow{\$} \mathbb{Z}_q} \left[\mathcal{A}^{\mathsf{DL}_{\mathsf{OM}}}(g^{k_0}, \ldots, g^{k_n}) = (k_0, \ldots, k_n)\right] \geq \varepsilon$$

where the probability is taken over the random tape of $\mathcal{A}$, $\mathcal{A}$ stops after time at most τ and $\mathcal{A}$ calls $\mathsf{DL}_{\mathsf{OM}}$ at most n times. The one-more discrete log assumption tells that no probabilistic algorithm can solve n-DL with non-negligible success probability over $\mathbb{G}$ for any integer $n \geq 1$. It is easily seen that DL is contained as the special case $\mathsf{DL} \equiv 0$-DL and that n_1-$\mathsf{DL} \Leftarrow n_2$-$\mathsf{DL}$ whenever $n_1 \geq n_2$.

2.4 Reduction-based Security Proofs

Reductions. Cryptographers use reductionist proofs to convince others that their schemes are computationally secure. An algorithm $\mathcal{R}$ is said to reduce a problem P_1 to a problem P_2, which we then denote by $P_1 \Leftarrow_{\mathcal{R}} P_2$, if $\mathcal{R}$ solves P_1 with the help of an algorithm solving P_2. Algorithm $\mathcal{R}$ is then called a *reduction* from P_1 to P_2. We write $P_1 \Leftarrow P_2$ when there exists a polynomial time reduction from P_1 to P_2, and $P_1 \equiv P_2$ when one has simultaneously $P_1 \Leftarrow P_2$ and $P_2 \Leftarrow P_1$.

Algebraic algorithms. Our method of converting a reduction $\mathcal{R}$ such that $\mathsf{DL} \Leftarrow_{\mathcal{R}} \mathsf{UF\text{-}KOA}\,[\Sigma_{\mathrm{H}}]$ into an algorithm solving the one-more discrete log problem applies whenever $\mathcal{R}$ belongs to a certain "natural" class of reductions. We refer to these as *algebraic reductions*.

In brief, a reduction algorithm $\mathcal{R}$ is *algebraic* with respect to a group $\mathbb{G}$ if $\mathcal{R}$ is limited to perform group operations on group elements. Adding $1_{\mathbb{G}}$ to $g \in \mathbb{G}$ is thus not permitted, even if this operation is well-defined and meaningful (if $\mathbb{G}$ is the multiplicative subgroup of a ring, for instance). $\mathcal{R}$ is free to apply arbitrary operations on other data types, but when it comes to elements of $\mathbb{G}$, the only available operations are among the (redundant) limited set

$$S = \{(g_1, g_2) \mapsto g_1 \stackrel{?}{=} g_2, (g_1, g_2) \mapsto g_1 \cdot g_2, (g_1, \lambda) \mapsto g_1^{\lambda}, g_1 \mapsto g_1^{-1}\} .$$

For instance a reduction placed into the generic group model (GGM) or more precisely in the non-programmable GGM is an algebraic reduction[4]. However, the class of algebraic reductions encompasses *much more* algorithms and in particular may be relevant on groups where there do exist algorithms exploiting the encoding of elements. This class of reductions is not overly restrictive (in fact, we do not know any example of a cryptographic reduction which is not algebraic). The restriction of our results to algebraic reductions is far much weaker than the one made in [11] which considers only reductions supplying the adversary with a public key which is always the same as its own challenge. It is worth noting that our results extend readily to such reductions.

Algebraic algorithms were originally defined by Boneh and Venkatesan [5] in the context of rings of integers modulo $n = pq$ under the form of straight-line programs computing polynomials over the ring structure $\mathbb{Z}_n$. Here, we stick to a (somewhat more natural) definition of algebraicity towards a group structure. A formal definition of this property is that an algebraic algorithm $\mathcal{R}$ admits a polynomial time extractor $\mathsf{Extract}$ enabling one, given $\mathcal{R}$'s inputs $(s, g_1, \ldots, g_k) \in \{0,1\}^* \times \mathbb{G}^k$ and random tape ϖ, to recover for any variable $h \in \mathbb{G}$ output by $\mathcal{R}$ after τ elementary steps, the coefficients α_i such that $h = g_1^{\alpha_1} \ldots g_k^{\alpha_k}$. $\mathsf{Extract}$ possibly has non black-box access to $\mathcal{R}$ and in particular may be given the code of $\mathcal{R}$. We require that $\mathsf{Extract}$ runs in time $\mathsf{poly}\,(\tau, |\mathcal{R}|)$ where $|\mathcal{R}|$ denotes the code size of $\mathcal{R}$.

In the sequel, we adopt the notation $P_1 \Leftarrow_{\mathrm{ALG}} P_2$ whenever there exists an algebraic algorithm $\mathcal{R}$ such that $P_1 \Leftarrow_{\mathcal{R}} P_2$ and $P_1 \equiv_{\mathrm{ALG}} P_2$ when $P_1 \Leftarrow_{\mathcal{R}_1} P_2$ and $P_2 \Leftarrow_{\mathcal{R}_2} P_1$ for algebraic reductions $\mathcal{R}_1, \mathcal{R}_2$. Conversely, the notation $P_1 \not\Leftarrow_{\mathrm{ALG}} P_2$ says that there exists no algebraic algorithm $\mathcal{R}$ such that $P_1 \Leftarrow_{\mathcal{R}} P_2$. We define $P_1 \not\equiv_{\mathrm{ALG}} P_2$ in a similar way.

[4] It should be mentioned that the GGM suffers from the same separation problems as the ROM, see [10].

3 Schnorr is Unbreakable under the One-More Discrete Log Assumption

We start by showing that Schnorr's signature scheme Σ_H defined over some group $\mathbb{G}$ is at least as hard to break that the one-more discrete log problem is hard to solve over $\mathbb{G}$. This is a *positive* security result standing in the standard model.

Theorem 1 (q_s-$\mathsf{DL} \Leftarrow \mathsf{BK\text{-}CMA}\,[\Sigma_{\mathrm{H}}]$). *Assume there exists an adversary $\mathcal{A}$ against Σ_H that breaks the secret key under a chosen-message attack with q_s signature queries and success probability ε. Then there exists an algorithm $\mathcal{R}$ that solves q_s-DL with probability $\varepsilon' = \varepsilon$ in similar time.*

Proof. The description of $\mathbb{G} = \langle g \rangle$ is implicitly given to all parties (this will be the case for all reductions and meta-reductions considered in this paper). Assume there exists a probabilistic algorithm $\mathcal{A}$ that takes as input $y = g^x$, requests the signature of q_s messages, and outputs the secret key x with probability ε after τ steps. We construct a reduction algorithm $\mathcal{R}$ which makes use of $\mathcal{A}$ to solve a q_s-DL instance over $\mathbb{G}$. Algorithm $\mathcal{R}$ works as follows.

$\mathcal{R}$ receives $q_s + 1$ group elements $r_0, \ldots, r_{q_s}$, defines $y = r_0$ and launches $\mathcal{A}(y, \varpi)$ over some random tape ϖ. Now whenever $\mathcal{A}$ requests the Schnorr signature of a message m_i, $\mathcal{R}$ uses r_i to compute $c_i = H(m_i, r_i)$. $\mathcal{R}$ then queries the discrete log oracle to get $s_i \leftarrow \mathsf{DL}_{\mathsf{OM}}(r_i \cdot y^{c_i})$ and returns the signature $\sigma_i = (s_i, c_i)$. It is easily seen that this simulation is perfect.

After at most q_s signature queries, $\mathcal{A}$ returns k_0 such that $r_0 = g^{k_0}$ with probability ε in which case $\mathcal{R}$ uses k_0 to retrieve the discrete logarithm $k_i = s_i - k_0 c_i \bmod q$ of r_i for $i = 1, \ldots, q_s$. $\mathcal{R}$ then returns $(k_0, k_1, \ldots, k_{q_s})$ and therefore succeeds in solving q_s-DL with probability $\varepsilon' = \varepsilon$ after at most $\tau' = \tau + \mathsf{poly}(q_s, \mathsf{Time}(H), \log q)$ steps. □

4 Schnorr Signatures are not Unforgeable under the Discrete Log Assumption

We now show that Schnorr signatures cannot be proven universally unforgeable under the discrete log assumption in the standard model with respect to an algebraic reduction. We actually show that if such a reduction existed then the one-more discrete log assumption would not hold over $\mathbb{G}$.

Theorem 2. *Assume that the one-more discrete log assumption holds. Then* $\mathsf{DL} \not\Leftarrow_{\mathrm{ALG}} \mathsf{UF\text{-}KOA}\,[\Sigma_{\mathrm{H}}]$.

We give a more precise formulation of Theorem 2 in the following lemma.

Lemma 1. *Assume there exists an algebraic reduction algorithm $\mathcal{R}$ that converts an (ε, τ)-universal forger $\mathcal{A}$ under a key-only attack into an (ε', τ')-solver for the discrete logarithm and assume that $\mathcal{R}$ executes $\mathcal{A}$ at most n times. Then there exists a meta-reduction algorithm $\mathcal{M}$ that solves n-DL with success probability $\varepsilon'' = \varepsilon'$ within time $\tau'' = \tau' + \mathsf{poly}(\tau', |\mathcal{R}|, n, \mathsf{Time}(H), \log q)$.*

Proof. The rest of the section is dedicated to proving Lemma 1 and we start by giving an overview of how the proof works. Assuming the existence of an algebraic reduction $\mathcal{R}$ as above, we construct a meta-reduction $\mathcal{M}$ that solves n-DL with success probability identical to the one of $\mathcal{R}$. Algorithm $\mathcal{M}$ works as follows. Given $n+1$ group elements $r_0, \ldots, r_n \in \mathbb{G}$, $\mathcal{M}$ launches $\mathcal{R}$ over r_0 and some arbitrary random tape. $\mathcal{M}$ then perfectly simulates at most n executions of the adversary $\mathcal{A}$ by using $r_1, \ldots, r_n$ and by making requests of discrete logarithms to oracle $\mathsf{DL}_{\mathsf{OM}}$. If $\mathcal{R}$ outputs k_0, $\mathcal{M}$ uses its transcript information to retrieve the discrete logs k_j of the r_j's.

Tracing $\mathcal{R}$'s internal group operations. The reduction algorithm $\mathcal{R}$ takes as input a challenge discrete log instance $r_0 = g^{k_0}$ and is allowed to invoke n times the universal forger $\mathcal{A}$ with freely chosen public keys $y_i = g^{x_i}$, messages m_i and random tapes ϖ_i where $i = 1, \ldots, n$. For our meta-reduction $\mathcal{M}$ to work, however, we must dispose of a constructive way to recover the value of the x_i's from the one of $k_0 = \mathsf{dl}_g(r_0)$. This is where an additional mechanism is needed. We may either choose to dive $\mathcal{R}$ into the generic model to have access to its internal computations involving group elements, or more generally consider $\mathcal{R}$ to be algebraic and let $\mathcal{M}$ dispose of the code of $\mathcal{R}$ if necessary, *i.e.* have non black-box access to $\mathcal{R}$. In the sequel, we impose that $\mathcal{R}$ is algebraic, and provided that the code of $\mathcal{R}$ is polynomial in length, $\mathcal{M}$ is assumed to dispose of a polynomial time extraction procedure $\mathsf{Extract}(k_0, \texttt{Transcript}) = (x_1, \ldots, x_n)$.

Simulation of $\mathcal{A}$. The simulation of a universal forger $\mathcal{A}(y, m, \varpi)$ under a key-only attack is described as follows. `Transcript` and j are viewed as global variables initialized before $\mathcal{A}$ is executed for the first time.

```
1. Receive $(y, m, \varpi) \in \mathbb{G} \times \{0,1\}^\star \times \{0,1\}^\star$
2. Select $\delta \xleftarrow{\$} [0,1]$ uniformly at random
3. If $\delta > \varepsilon$ stop and output $\perp$
4. Else if $(y, m, \varpi) \mapsto (s, c) \in$ Transcript for some signature $(s, c)$, stop and
   output $(s, c)$
5. Else
   (a) Define $r = r_j$ and increment $j$ by 1
   (b) Compute $c = H(m, r)$
   (c) Request the discrete log $s \leftarrow \mathsf{DL}_{\mathsf{OM}}(ry^c)$
   (d) Append $(y, m, \varpi) \mapsto (s, c)$ to Transcript
   (e) Output $\sigma = (s, c)$
```

Description of $\mathcal{M}$. $\mathcal{M}$ takes the first group element $r_0 \in \mathbb{G}$, initializes $j = 1$ and $\texttt{Transcript} = \emptyset$, and invokes $\mathcal{R}$ with input r_0 and arbitrary random tape. $\mathcal{M}$ then simulates the universal forger $\mathcal{A}$ as above, resulting in a perfect simulation. During simulation, $\mathcal{M}$ sends ℓ requests to oracle $\mathsf{DL}_{\mathsf{OM}}$ for some $\ell \in [1, n]$ (therefore ℓ is the value of j after the n successive simulations of $\mathcal{A}$). Now assume $\mathcal{R}$ outputs $k_0 = \mathsf{dl}_g(r_0)$. $\mathcal{M}$ then uses its transcript information to extract

$$(x_1, \ldots, x_n) = \mathsf{Extract}(k_0, \texttt{Transcript}) \;.$$

There are ℓ records of type $(y, m, \varpi) \mapsto (s, c)$ in `Transcript`. Then for $j \in [1, \ell]$, if the j-th record is of the form $(g^{x_i}, *, *) \mapsto (s, c)$ for some $i \in [1, n]$ then $\mathcal{M}$ computes $k_j = \mathsf{dl}_g(r_j) = s - cx_i \bmod q$. At this point, $\mathcal{M}$ knows $(k_0, k_1, \ldots, k_\ell)$. Now for $j = \ell+1$ to n, $\mathcal{M}$ directly requests $k_j = \mathsf{dl}_g(r_j)$ to $\mathsf{DL}_{\mathsf{OM}}$. $\mathcal{M}$ then returns $(k_0, \ldots, k_\ell, k_{\ell+1}, \ldots, k_n)$, thereby succeeding in solving n-DL. This occurs with probability $\varepsilon'' = \varepsilon'$ and time $\tau'' = \tau' + \mathsf{poly}(\tau', |\mathcal{R}|, n, \mathsf{Time}(H), \log q)$. □

5 Extension to the One-More Discrete Log Assumption

Theorem 2 shows that under the one-more discrete log assumption, no algebraic reduction exists that would reduce the discrete log problem to forging Schnorr signatures. This is a big step towards proving that coming up with forgeries is strictly easier than extracting discrete logs. One may ask whether a similar impossibility result extends to computational problems weaker than DL. We provide a positive answer to this question too by showing that if the one-more discrete log assumption holds, there can be no algebraic reduction from solving *any* one-more discrete log problem to forging signatures. In other words

Theorem 3. *Assume that the one-more discrete log assumption holds. Then*

$$t\text{-}\mathsf{DL} \quad \not\Leftarrow_{\mathrm{ALG}} \quad \mathsf{UF\text{-}KOA}[\Sigma_{\mathrm{H}}]$$

for any integer $t \geq 0$.

Note that Theorem 3 contains Theorem 2 in the special case where $t = 0$. This shows that Schnorr signatures cannot be proven universally unforgeable under the one-more discrete log assumption with respect to an algebraic reduction, or that if they can, the one-more discrete log assumption does not hold over $\mathbb{G}$, thus rendering such a reduction useless. The following lemma captures this more precisely.

Lemma 2. *Assume there exists an algebraic reduction algorithm $\mathcal{R}$ that converts an (ε, τ)-universal forger $\mathcal{A}$ under a key-only attack into an (ε', τ')-solver for t-DL and assume that $\mathcal{R}$ executes $\mathcal{A}$ at most n times. Then there exists a meta-reduction algorithm $\mathcal{M}$ that solves $(t+n)$-DL with success probability $\varepsilon'' = \varepsilon'$ within time $\tau'' = \tau' + \mathsf{poly}(\tau', |\mathcal{R}|, t, n, \mathsf{Time}(H), \log q)$.*

Proof (of Lemma 2). The proof is very similar to the one of Lemma 1. We therefore avoid details and focus on the changes we apply to extend to the general case t-DL, $t \geq 0$. Again, from an algebraic reduction $\mathcal{R}$ as above, we construct $\mathcal{M}$ that solves $(t+n)$-DL with success probability identical to the one of $\mathcal{R}$.

Extraction of secret keys. The reduction algorithm $\mathcal{R}$ now takes as input a t-DL instance

$$\left(r_0 = g^{k_0}, r_1 = g^{k_1}, \ldots, r_t = g^{k_t}\right) \in \mathbb{G}^{t+1} ,$$

calls $\mathsf{DL_{OM}}$ up to t times and invokes at most n times the universal forger $\mathcal{A}$ with freely chosen public keys $y_i = g^{x_i}$, messages m_i and random tapes ϖ_i where $i = 1, \ldots, n$. Since $\mathcal{R}$ is algebraic and of polynomially bounded size, we dispose of a polynomial time extraction procedure $\mathsf{Extract}(k_0, k_1, \ldots, k_t, \mathtt{Transcript}) = (x_1, \ldots, x_n)$.

Simulation of $\mathcal{A}$. The simulation of the universal forger $\mathcal{A}$ is identical to the one given in the previous section.

Simulation of $\mathsf{DL_{OM}}$. Since $\mathcal{R}$ attempts to solve t-DL, we must allow $\mathcal{R}$ to send up to t requests to the discrete logarithm oracle $\mathsf{DL_{OM}}$. The meta-reduction $\mathcal{M}$ individually collects these requests, forwards them to $\mathsf{DL_{OM}}$ and sends the corresponding outputs back to $\mathcal{R}$. We may assume that $\mathcal{R}$ makes exactly t oracle calls since in the case when $\mathcal{R}$ sends strictly less than t requests during the game, $\mathcal{M}$ sends additional requests of discrete logs for randomly chosen group elements to $\mathsf{DL_{OM}}$ on behalf of $\mathcal{R}$. This simulation is obviously perfect.

Overall description of $\mathcal{M}$. $\mathcal{M}$ takes its first $t+1$ group elements $(r_0, \ldots, r_t)$ among $(r_0, \ldots, r_{t+n})$, initializes $\mathtt{Transcript} = \emptyset$ and $j = 1$, and invokes $\mathcal{R}$ with input $(r_0, \ldots, r_t)$ and arbitrary random tape. $\mathcal{M}$ then simulates the universal forger $\mathcal{A}$ and discrete log oracle $\mathsf{DL_{OM}}$ as above, resulting in a perfect simulation. During simulation, $\mathcal{M}$ sends $t + \ell$ requests to $\mathsf{DL_{OM}}$ for some $\ell \in [1, n]$. Now assume $\mathcal{R}$ succeeds and outputs

$$k_0 = \mathsf{dl}_g(r_0), k_1 = \mathsf{dl}_g(r_1), \ldots, k_t = \mathsf{dl}_g(r_t) \ .$$

$\mathcal{M}$ then uses its transcript information to extract

$$(x_1, \ldots, x_n) = \mathsf{Extract}(k_0, \ldots, k_t, \mathtt{Transcript}) \ .$$

There are ℓ records of type $(y, m, \varpi) \mapsto (s, c)$ in $\mathtt{Transcript}$. Then for $j \in [1, \ell]$, if the j-th record is of the form $(g^{x_i}, *, *) \mapsto (s, c)$ for some $i \in [1, n]$ then $\mathcal{M}$ computes $k_{t+j} = \mathsf{dl}_g(r_{t+j}) = s - cx_i \bmod q$. Thus $\mathcal{M}$ recovers $(k_{t+1}, \ldots, k_{t+\ell})$. Now for $j = t + \ell + 1$ to n, $\mathcal{M}$ directly requests $k_j = \mathsf{dl}_g(r_j)$ to $\mathsf{DL_{OM}}$. Then $\mathcal{M}$ returns

$$\underbrace{(k_0, k_1, \ldots, k_t)}_{\text{ouput by } \mathcal{R}} \cup \underbrace{(k_{t+1}, \ldots, k_{t+\ell})}_{\text{extracted by } \mathcal{M}} \cup \underbrace{(k_{t+\ell+1}, \ldots, k_{t+n})}_{\text{requested to } \mathsf{DL_{OM}}} = (k_0, \ldots, k_{t+n}) \ ,$$

thereby succeeding in solving $(t + n)$-DL. This occurs with probability $\varepsilon'' = \varepsilon'$ and execution time $\tau'' = \tau' + \mathsf{poly}(\tau', |\mathcal{R}|, t, n, \mathsf{Time}(H), \log q)$. □

Summary. Because of the relations

$$\mathsf{EF\text{-}CMA}[\Sigma_\mathrm{H}] \Leftarrow \{\ \mathsf{EF\text{-}KOA}[\Sigma_\mathrm{H}], \mathsf{UF\text{-}CMA}[\Sigma_\mathrm{H}]\ \} \Leftarrow \mathsf{UF\text{-}KOA}[\Sigma_\mathrm{H}] \ ,$$

our impossibility results readily extend to forgeries of any kind, under any attack model. We summarize our results (also displayed on Figure 2), stating our positive and negative security proofs for Schnorr signatures assuming the one-more discrete log assumption holds:

Goal vs. Atk	**Key-Only Attacks**	**Chosen-Message Attacks**
Forgeries	$\boxed{\not\equiv_{\mathrm{ALG}}\mathsf{DL} \text{ and even } \not\equiv_{\mathrm{ALG}} t\text{-}\mathsf{DL} \text{ for } t > 0}$	
Breakability	$\equiv \mathsf{DL}$	$\boxed{\Rightarrow q_s\text{-}\mathsf{DL}}$

Fig. 2. Our results for Schnorr's signature scheme Σ_H are shown in boxes. In particular, universal and existential forgeries under any kind of attack cannot be proven equivalent to the discrete log problem via an algebraic reduction.

Theorem 1: Schnorr's scheme is unbreakable under chosen-message attacks.

Theorems 2 and 3: Universal and existential forgeries under any kind of attack cannot be proven secure under the discrete log assumption or even the one-more discrete log assumption with respect to an algebraic reduction.

6 Applications to Other Signature Schemes

We extend our results to various signature schemes, adapting our meta-reduction-based proof technique to comply with the schemes' inner design.

6.1 Guillou-Quisquater

GQ signatures were suggested by Guillou and Quisquater in [15]. Among other properties, GQ is a Fiat-Shamir-transformed signature scheme based on RSA and supports identity-based public keys.

Scheme parameters and key generation. Let p, q be two large primes, set $n = pq$ and choose randomly v such that $\gcd(v, \phi(n)) = 1$. The public parameters are (n, v) as well as a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_v$. Now the signer chooses a secret key $x \xleftarrow{\$} \mathbb{Z}_n$. The related public key is $y = x^{-v} \bmod n$.

Signature generation and verification. Given a message m, the signer selects $k \xleftarrow{\$} \mathbb{Z}_n$, computes $r = k^v \bmod n$, $c = H(m, r)$ and $s = kx^c \bmod n$. The signature is $\sigma = (s, c)$. To verify the signature, check whether $H(m, s^v y^c \bmod n) = c$.

Because of their similarity with Schnorr, GQ signatures fit our impossibility proofs quite well. However the primitive computational problem here is not DL but rather extracting v-th roots modulo n, which we denote of course by RSA. The one-more version of RSA is easily defined with the help of an oracle $\mathsf{RSA}_{\mathsf{OM}}$ extracting the v-th root of its argument [3]. Solving n-RSA thus consists in finding the v-th root of $n+1$ elements of $\mathbb{Z}_n$ given no more than n invocations of $\mathsf{RSA}_{\mathsf{OM}}$. The one-more RSA assumption says that n-RSA is intractable for $n \geq 1$.

Theorem 4. *Assume the one-more RSA assumption holds. Then (i) GQ is unbreakable under chosen-message attacks. (ii) Universal and existential forgeries* *under any attack cannot be proven secure under the RSA assumption or the one-more RSA assumption with respect to an algebraic reduction.*

Proof (Sketch). We rely on the same proof technique as in the proofs of Theorems 1, 2 and 3. Here, however, the simulation of the $\mathsf{UF\text{-}KOA}$ adversary $\mathcal{A}$ must be slightly reformulated. An overall description of our meta-reduction $\mathcal{M}$ is as follows. The reduction algorithm $\mathcal{R}$ takes as input a t-RSA instance

$$(r_0 = k_0^v \bmod n, r_1 = k_1^v \bmod n, \ldots, r_t = k_t^v \bmod n) \in \mathbb{Z}_n^{t+1} ,$$

calls $\mathsf{RSA_{OM}}$ up to t times and calls the forger $\mathcal{A}$ at most n times with public keys $y_i = {x_i}^{-v} \bmod n$, messages m_i and random tapes ϖ_i where $i = 1, \ldots, n$. Since $\mathcal{R}$ is algebraic, $\mathcal{M}$ is assumed to dispose of a polynomial time extraction procedure $\mathsf{Extract}(k_0, k_1, \ldots, k_t, \mathtt{Transcript}) = (x_1, \ldots, x_n)$. Now when simulating $\mathcal{A}(y, m, \varpi)$ for new inputs (y, m, ϖ), if $\mathcal{M}$ must compute a forgery then $\mathcal{M}$ takes $r = r_j$, computes $c = H(m, r)$ and requests the v-th root $s \leftarrow \mathsf{RSA_{OM}}(ry^{-c} \bmod n)$. The simulation is perfect. After recovering $(x_1, \ldots, x_n)$ from $(k_0, \ldots, k_t)$, $\mathcal{M}$ consults its transcript and if the j-th entry is $(x_i^{-v} \bmod n, *, *) \mapsto (s, c)$ for some i then $\mathcal{M}$ computes $k_{t+j} = sx_i^c = \sqrt[v]{r_{t+j}} \bmod n$. The unused inputs $r_{t+\ell+1}, \ldots, r_{t+n}$ are sent by $\mathcal{M}$ to $\mathsf{RSA_{OM}}$ to retrieve their v-th root directly. Following this slightly modified description of $\mathcal{M}$, one gets as before $\varepsilon'' = \varepsilon'$ and $\tau'' = \tau' + \mathsf{poly}(\tau', |\mathcal{R}|, t, n, \mathsf{Time}(H), \log q)$. $\square$

6.2 DSA, ECDSA and Generic DSA

DSA is a signature scheme standardized by the NIST in 1991 [9]. The original version of DSA is based on the discrete log problem over the subgroup of $\mathbb{Z}_p^*$ of prime order $q|p-1$. ECDSA, standardized as well [1], presents the same structure but is defined over a prime-order subgroup of an elliptic curve. We consider here their generalization to arbitrary prime-order groups as suggested by Brown in [7].

Scheme parameters and key generation. Again, $\mathbb{G} = \langle g \rangle$ denotes a group of prime order q. The public parameters are $(\mathbb{G}, g)$, a function $G : \mathbb{G} \mapsto \mathbb{Z}_q$ and a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$. The signer chooses a secret key $x \xleftarrow{\$} \mathbb{Z}_q$. The related public key is $y = g^x \in \mathbb{G}$.

Signature generation and verification. Given a message m, the signer selects $k \xleftarrow{\$} \mathbb{Z}_q^*$, computes $r = g^k$, $\rho = G(r)$, $u = H(m)$ and $s = k^{-1}(u + \rho x) \bmod q$. The signature is $\sigma = (\rho, s)$. To verify the signature, check whether $G\left(g^{H(m)/s} \cdot y^{\rho/s}\right) = \rho$.

Note that the original DSA corresponds to the case where $\mathbb{G} = \left(\mathbb{Z}_p^*\right)^{(p-1)/q}$, $|q| = 160$, $H = $ SHA-1 and $G(r) = r \bmod q$. Let E be an elliptic curve group over a finite field admitting an element P of prime order q with $|q| = 160$. ECDSA is obtained with $g = P$, $\mathbb{G} = \langle g \rangle$, $H = $ SHA-1 and $G(r) = x_r \bmod q$ where x_r is an integer representation of the x-coordinate of point r.

Before stating our security results, we define a variant of the one-more discrete log problem n-DL as follows. n-$\mathsf{DL}^\star$ consists in computing the discrete logs with respect to a fixed base g of $n+1$ group elements with bounded (to n) access to a discrete log oracle $\mathsf{DL}^\star_{\mathsf{OM}}$. Unlike $\mathsf{DL}_{\mathsf{OM}}$ which was limited to the fixed base g, $\mathsf{DL}^\star_{\mathsf{OM}}$ provides discrete logarithms with respect to any base $h \in \mathbb{G}$ meaning that $\mathsf{DL}^\star_{\mathsf{OM}}(h^\alpha, h)$ returns α for any $h \in \mathbb{G}$. Although 0-$\mathsf{DL}^\star \equiv \mathsf{DL} \equiv 0$-$\mathsf{DL}$, one only has in the general[5] case n-$\mathsf{DL}^\star \Leftarrow n$-$\mathsf{DL}$ for $n \geq 1$. The *one-more free-base discrete log assumption* says that n-$\mathsf{DL}^\star$ is intractable for $n \geq 1$.

Theorem 5. *Assume the one-more free-base discrete log assumption holds. Then (i) Generic DSA is unbreakable under chosen-message attacks. (ii) Universal and existential forgeries under any attack cannot be proven secure under the discrete log assumption or the one-more free-base discrete log assumption with respect to an algebraic reduction.*

Proof (Sketch). We use the same proof technique as for Theorem 1 and Lemmas 1 and 2. What we are after is a reduction q_s-$\mathsf{DL}^\star \Leftarrow \mathsf{BK}\text{-}\mathsf{CMA}$ [Generic-DSA] as well as a means to simulate an $\mathsf{UF}\text{-}\mathsf{KOA}$ adversary $\mathcal{A}$ leading to a meta-reduction $\mathcal{M}$ such that if t-$\mathsf{DL}^\star \Leftarrow_{\mathcal{R}} \mathsf{UF}\text{-}\mathsf{KOA}$ [Generic-DSA] where $\mathcal{R}$ is limited to n executions of $\mathsf{UF}\text{-}\mathsf{KOA}$ [Generic-DSA], then n-$\mathsf{DL}^\star \Leftarrow_{\mathcal{M}} \mathcal{R}$. We first have to show how to simulate a signing oracle without knowing the secret key. Remembering that the simulator is given group elements $\{r_j\}$ for $j \in [1, q_s]$ or $[t+1, t+n]$, the signature simulation is as follows. For a given public key $y = g^x \in \mathbb{G}$ and a message m, we define $r = r_j$ and compute $\rho = G(r)$ and $u = H(m)$. We then invoke $\mathsf{DL}^\star_{\mathsf{OM}}$ to get

$$s = \mathsf{DL}^\star_{\mathsf{OM}}\left(g^u \cdot y^\rho, r\right) \ .$$

It is easy to see that if we write $r = g^k$ then s conforms to the equation $s = k^{-1}(u + \rho x) \bmod q$. The simulator then outputs $\sigma = (\rho, s)$. The simulation is obviously perfect. We now have to show how to recover $k_j = \mathsf{dl}_g(r_j)$ from *a)* either the list of secret keys $\{x_i\}$ given to simulation number $i \in [1, q_s]$ or $[1, n]$ *b)* or from the outputs $k_0, \ldots, k_t$ of $\mathcal{R}$. Since $\mathcal{R}$ is algebraic, the key extraction procedure using `Transcript` leads case *b)* to case *a)*. Therefore, we are left with the task of recovering k_j from x_i and the transcript of our simulations. This is easily done by inverting the signature formula to recover $k_j = s^{-1}(u + \rho x_i) \bmod q$. □

6.3 KCDSA and Trusted ElGamal Signatures Type I

DSA and DSA-like signature schemes have been extended in many ways. We focus on a generalization called TEGTSS-I put forward by Brickell *et al.* in [6]. This extension contains the korean standard KCDSA [17] as a particular case.

Scheme parameters and key generation. Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q. Now define three functions $f_1 : \mathbb{Z}_q^4 \mapsto \mathbb{Z}_q$, $f_2 : \mathbb{Z}_q^3 \mapsto \mathbb{Z}_q$ and $f_3 : \mathbb{Z}_q^3 \mapsto \mathbb{Z}_q$

[5] The converse is unknown.

such that for any integers $k, x, u, \rho \in \mathbb{Z}_q$,

$$\text{if } s = f_1(k, x, u, \rho) \text{ then } f_2(s, u, \rho) + x f_3(s, u, \rho) \equiv k \mod q \ .$$

The public parameters are $(\mathbb{G}, g, f_1, f_2, f_3)$, a function $G : \mathbb{G} \mapsto \mathbb{Z}_q$ and a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$. The signer chooses a secret key $x \xleftarrow{\$} \mathbb{Z}_q$. The related public key is $y = g^x \in \mathbb{G}$.

Signature generation and verification. Given a message m, the signer selects $k \xleftarrow{\$} \mathbb{Z}_q^*$, computes $r = g^k$, $\rho = G(r)$, $u = H(m)$ and $s = f_1(k, x, u, \rho)$. The signature is $\sigma = (\rho, s)$. To verify the signature, compute $u = H(m)$, $\alpha = f_2(s, u, \rho)$, $\beta = f_3(s, u, \rho)$ and check whether $G\left(g^\alpha \cdot y^\beta\right) = \rho$.

KCDSA fulfils this description where $\mathbb{G} = \left(\mathbb{Z}_p^*\right)^{(p-1)/q} = \langle g \rangle$, H and G are hash functions mapping $\mathbb{Z}_p$ to $\mathbb{Z}_q$, and functions f_1, f_2, f_3 are defined by

$$\begin{aligned} f_1(k, x, u, \rho) &= (k - u \oplus \rho)/x \mod q \ , \\ f_2(s, u, \rho) &= u \oplus \rho \ , \\ f_3(s, u, \rho) &= s \ . \end{aligned}$$

Before stating any security property of TEGTSS-I signatures, we leave as an exercise to the reader to prove the following property.

Claim. Let f_1, f_2, f_3 be functions as above. Then there exist efficiently computable functions $\delta_1, \delta_2, \delta_3, \delta_4$ and ϵ mapping $\mathbb{Z}_q^2$ to $\mathbb{Z}_q$ and such that $\delta_1(u, \rho) \neq 0$, $\delta_3(u, \rho) \cdot \delta_4(u, \rho) \neq 0$, $\epsilon(u, \rho) \neq 0$ for any $u, \rho \in \mathbb{Z}_q$ and

$$f_1(k, x, u, \rho) = \left(\frac{\delta_1(u, \rho)k + \delta_2(u, \rho)}{\delta_3(u, \rho)x + \delta_4(u, \rho)}\right)^{\frac{1}{\epsilon(u,\rho)}} \ , \tag{1}$$

$$f_2(s, u, \rho) = \frac{\delta_4(u, \rho) s^{\epsilon(u,\rho)} - \delta_2(u, \rho)}{\delta_1(u, \rho)} \ , \tag{2}$$

$$f_3(s, u, \rho) = \frac{\delta_3(u, \rho) s^{\epsilon(u,\rho)}}{\delta_1(u, \rho)} \ , \tag{3}$$

where all evaluations are modulo q.

As an illustration, KCDSA yields $\delta_1(u, \rho) = 1$, $\delta_2(u, \rho) = -u \oplus \rho$, $\delta_3(u, \rho) = 1$, $\delta_4(u, \rho) = 0$ and $\epsilon(u, \rho) = 1$. Note that DSA is also a particular case if we set $\delta_1(u, \rho) = 1$, $\delta_2(u, \rho) = 0$, $\delta_3(u, \rho) = \rho$, $\delta_4(u, \rho) = u$ and $\epsilon(u, \rho) = -1$. We now state our security results. As for Generic DSA, we rely on n-$\mathsf{DL}^\star$ and the one-more free-base discrete log assumption:

Theorem 6. *Let Σ be a signature scheme of type TEGTSS-I. Assume the one-more free-base discrete log assumption holds. Then (i) Σ is unbreakable under chosen-message attacks. (ii) Universal and existential forgeries under any attack cannot be proven secure under the discrete log assumption or the one-more free-base discrete log assumption with respect to an algebraic reduction.*

Proof (Sketch). Here again, we make use of the proofs of Theorem 1, Lemmas 1 and 2. As discussed earlier, it is necessary to show how to simulate a signing oracle without knowing the secret key. Recall the simulator is given group elements $\{r_j\}$ for $j \in [1, q_s]$ or $[t+1, t+n]$. Now the signature simulation is as follows. For a given public key $y = g^x \in \mathbb{G}$ and a message m, we define $r = r_j$ and compute $\rho = G(r)$ and $u = H(m)$. Using our claim above, we compute $\delta_i = \delta_i(u, \rho)$ for $i \in [1, 4]$ and then invoke $\mathsf{DL}^{\star}_{\mathsf{OM}}$ to get

$$s = \left(\mathsf{DL}^{\star}_{\mathsf{OM}}\left(g^{\delta_4(u,\rho)} \cdot y^{\delta_3(u,\rho)}, r^{\delta_1(u,\rho)} \cdot g^{\delta_2(u,\rho)}\right)\right)^{\frac{1}{\epsilon(u,\rho)}} .$$

Now writing $r = g^k$, we easily see that s conforms to the signature equation $s = f_1(k, x, u, \rho) \bmod q$. The simulator then outputs $\sigma = (\rho, s)$ and the simulation is perfect. Following the same argument as in the proof of Theorem 5, we now have to show how to recover k_j from x_i and the transcript of our simulations. This directly follows from the definition of TEGTSS-I since $k_j = f_2(s, u, \rho) + x_i \cdot f_3(s, u, \rho) \bmod q$. □

6.4 Trusted ElGamal Signatures Type II

Trusted ElGamal signatures of type II form another family of discrete-log-based signatures and were also suggested by Brickell *et al.* in [6]. TEGTSS-II are similar to TEGTSS-I in that functions f_1, f_2, f_3 are defined along the same lines and the generation of the public parameters and user keys is identical.

Signature generation and verification. Given a message m, the signer selects $k \xleftarrow{\$} \mathbb{Z}_q^*$, computes $r = g^k$, $\rho = G(r)$, $u = H(m, \rho)$ and $s = f_1(k, x, u, \rho)$. The signature is $\sigma = (\rho, s)$. To verify the signature, compute $u = H(m, \rho)$, $\alpha = f_2(s, u, \rho)$, $\beta = f_3(s, u, \rho)$ and check whether $G\left(g^\alpha \cdot y^\beta\right) = \rho$.

Therefore, TEGTSS-II signatures define $u = H(m, \rho)$ instead of $u = H(m)$ while generating or verifying the signature. It is straightforward that Theorem 6 still applies in this case. The proof is identical except that the signature simulator now defines $r = r_j$ and computes $\rho = G(r)$ and $u = H(m, \rho)$.

6.5 ElGamal and Meta-ElGamal Signatures

ElGamal signatures were suggested in 1984 [12] and generalized later by Horster, Michels and Petersen [16]. We consider here a similar generalization to arbitrary prime-order groups.

Scheme parameters and key generation. Let $\mathbb{G} = \langle g \rangle$ be a group of prime order q. Define three functions $F_1, F_2, F_3 : \mathbb{G} \times \{0,1\}^* \times \mathbb{Z}_q$ such that $F_i(r, m, s)$ is linear in s for $i \in [1, 3]$. F_1, F_2 and F_3 may involve arbitrarily many hash functions. The public parameters are $(\mathbb{G}, g, F_1, F_2, F_3)$. The signer selects a secret key $x \xleftarrow{\$} \mathbb{Z}_q$. The public key is $y = g^x \in \mathbb{G}$.

Signature generation and verification. Given a message m, the signer selects $k \xleftarrow{\$} \mathbb{Z}_q^*$, computes $r = g^k$ and solves the linear equation

$$F_1(r, m, s) \equiv x \cdot F_2(r, m, s) + k \cdot F_3(r, m, s) \mod q \tag{4}$$

which solution is some $s \in \mathbb{Z}_q$. The signature is then $\sigma = (r, s) \in \mathbb{G} \times \mathbb{Z}_q$. To verify the signature, check whether

$$g^{F_1(r,m,s)} = y^{F_2(r,m,s)} \cdot r^{F_3(r,m,s)} \ .$$

Original ElGamal signatures define $\mathbb{G}$ as the subgroup of order $q|p-1$ of $\mathbb{Z}_p^*$, $F_1(r, m, s) = m$ or for long messages $F_1(r, m, s) = H(m)$ where H is a hash function mapping strings to $\mathbb{Z}_q$, $F_2(r, m, s) = r \bmod q$ and $F_3(r, m, s) = s$. We now give our results for any Meta-ElGamal scheme *i.e.* for any choice of F_1, F_2, F_3 as above. We still rely on n-$\mathsf{DL}^\star$ and the one-more free-base discrete log assumption.

Theorem 7. *Let Σ be a Meta-ElGamal signature scheme. Assume the one-more free-base discrete log assumption holds. Then (i) Σ is unbreakable under chosen-message attacks. (ii) Universal and existential forgeries under any attack cannot be proven secure under the discrete log assumption or the one-more free-base discrete log assumption with respect to an algebraic reduction.*

Proof (Sketch). As discussed above, it is enough to show how to simulate a signing oracle without knowing the secret key and recover k_j from x afterwards. Recalling that the simulator is given group elements $\{r_j\}$ for $j \in [1, q_s]$ or $[t+1, t+n]$, the signature simulation is as follows. For a given public key $y = g^x \in \mathbb{G}$ and a message m, we define $r = r_j$ and compute (as functions of m and r) the coefficients a_1, b_1, a_2, b_2, a_3 and b_3 such that $F_i(r, m, s) = a_i s + b_i$ for $i \in [1, 3]$. We then call $\mathsf{DL}^\star_{\mathsf{OM}}$ to get

$$s = \mathsf{DL}^\star_{\mathsf{OM}}\left(g^{a_1} y^{-a_2} r^{-a_3}, g^{-b_1} y^{b_2} r^{b_3}\right) \ .$$

Obviously, s conforms to the verification equation. The simulator then outputs $\sigma = (r, s)$ and the simulation is perfect. Now when $\mathcal{R}$ or $\mathcal{M}$ knows all the values of x, the transcript of the simulation involving r_j leads to specific values for (r, m, s). Then k_j is recovered as the unique solution in k of the signature equation Eq. 4. $\square$

7 Impossibility Results in the Random Oracle Model

All known reductions attesting the unforgeability of Fiat-Shamir-transformed signatures in the random oracle model lead to a loss factor close to q_H in terms of execution time or success probability [19]. Since a reasonable bound on the number of possible hash queries is around $q_H = 2^{80}$, this loss definitely makes these reductions loose, and subsequently imply larger keys and lowered performances. There exists no proof that this loss factor is necessary. The following theorem states however, that if the one-more discrete logarithm assumption holds then each and every algebraic reduction from computing the discrete logarithm to forging Schnorr signatures must lose at least a factor $\sqrt{q_H}$.

We note that a similar result can be extended to the one-more discrete log problems. Also, although we do not extend our work further in this direction, it is easily seen that this result applies to the random-oracle security of other signature schemes as well. We start by stating a few statistical facts.

Lemma 3 (Birthday paradox). *We consider an experiment in which n objects are drawn uniformly at random from a set of m elements. Then,*

1. *the probability of selecting the same element twice is*
$$P(m,n) = 1 - \frac{m(m-1)\dots(m-n+1)}{m^n}.$$
2. *when $n = O(\sqrt{m})$ and as $m \to \infty$, one gets*
$$P(m,n) \to 1 - \exp\left(-\frac{n(n-1)}{2m} + O\left(\frac{1}{\sqrt{m}}\right)\right) \simeq 1 - \exp\left(-\frac{n^2}{2m}\right).$$

Lemma 4. *Let q be a rational prime number, then*
$$|\mathrm{GL}_n(\mathbb{F}_q)| = (q^n-1)(q^n-q)(q^n-q^2)\dots(q^n-q^{n-1}) .$$
Therefore, the probability $z(n,q)$ that an $n \times n$ matrix picked at random is non-invertible is
$$z(n,q) = 1 - \frac{(q^n-1)(q^n-q)(q^n-q^2)\dots(q^n-q^{n-1})}{q^{n^2}} \le \frac{n}{q} .$$

Theorem 8. *Assume there exists an algebraic reduction algorithm $\mathcal{R}$ that converts an (ε, τ, q_H)-universal forger $\mathcal{A}$ under a key-only attack in the random oracle into an (ε', τ')-solver for the discrete logarithm and assume that $\mathcal{R}$ executes $\mathcal{A}$ at most n times. Then there exists a probabilistic algorithm $\mathcal{M}$ that solves n-DL with success probability $\varepsilon'' \ge \varepsilon' \cdot \exp\left(-\frac{n^2}{2q_H}\right) \cdot \left(1 - \frac{n}{q}\right)$ within time $\tau'' = \tau' + \mathsf{poly}\,(\tau', |\mathcal{R}|, n, q_H, \log q)$.*

Proof. Assuming the existence of an algebraic reduction $\mathcal{R}$ as above, we construct a meta-reduction $\mathcal{M}$ that solves n-DL. $\mathcal{R}$ takes as input a challenge discrete log instance $r_0 = g^{k_0}$ and is allowed to invoke n times the universal forger $\mathcal{A}$ with freely chosen public keys $y_i = g^{x_i}$, messages m_i and random tapes ϖ_i where $i = 1, \dots, n$. Without loss of generality, we may assume that the n invocations of $\mathcal{R}$, are pairwise distinct *i.e.* that two distinct executions of $\mathcal{A}$ differ in the value of the public key and/or the random tape, and/or at least one value returned by the random oracle H of $\mathcal{R}$.

Simulation of $\mathcal{A}$. $\mathcal{M}$ attempts to simulate at most n executions of the adversary $\mathcal{A}$ by using the vector of group elements $\boldsymbol{r} = (r_1, \dots, r_n)$ and by making requests to the discrete-log oracle $\mathsf{DL}_{\mathsf{OM}}$. More specifically, the i-th invocation of $\mathcal{A}$ is simulated as follows:

```
1. Receive $(y_i, m_i, \varpi_i) \in \mathbb{G} \times \{0,1\}^\star \times \{0,1\}^\star$
2. For $h \in [1, q_H]$
   (a) Randomly select $\boldsymbol{\alpha}_h \xleftarrow{\$} (\mathbb{Z}_q)^n$
   (b) Query $H$ to get $c_h = H\left(m_i, \boldsymbol{r}^{\boldsymbol{\alpha}_h}\right)$
3. Randomly select $\ell_i \xleftarrow{\$} [1, q_H]$
   (a) Set $c_i \leftarrow c_{\ell_i}$ and $\boldsymbol{\beta}_i \leftarrow \boldsymbol{\alpha}_{\ell_i}$
   (b) Request $s_i \leftarrow \mathsf{DL}_{\mathsf{OM}}\left(\boldsymbol{r}^{\boldsymbol{\beta}_i} \cdot y_i^{c_i}\right)$
   (c) Append $(y_i, m_i, \varpi_i) \mapsto (s_i, c_i)$ and $(\ell_i, \boldsymbol{\beta}_i)$ to Transcript
4. Pick at random $\delta \in [0,1]$
5. If $\delta > \varepsilon$ return $\perp$
6. Else return $\sigma_i = (s_i, c_i)$
```

Here, if $\boldsymbol{a} = (a_1, \ldots, a_w)$ and $\boldsymbol{b} = (b_1, \ldots, b_w)$ then $\boldsymbol{a}^{\boldsymbol{b}}$ stands for $\prod_{\kappa=1}^{w} a_\kappa^{b_\kappa}$. Note that all random selections made by $\mathcal{A}$ are in fact pseudo-random in ϖ_i and all hash values c_h defined by H when the selection takes place.

Extraction of discrete logs. Again, we assume that $\mathcal{M}$ disposes of a polynomial time extraction procedure $\mathsf{Extract}(k_0, \texttt{Transcript}) = (x_1, \ldots, x_n)$ *i.e.* we consider $\mathcal{R}$ to be algebraic. Therefore, if $\mathcal{R}$ outputs k_0, $\mathcal{M}$ uses its transcript information to retrieve the discrete logs x_i of the y_i's. Now $\mathcal{M}$ attempts to solve over $\mathbb{Z}_q$ the linear system

$$\begin{cases} \boldsymbol{\beta}_1 \cdot \boldsymbol{k} \equiv s_1 - c_1 \cdot x_1 \mod q \\ \quad\vdots \\ \boldsymbol{\beta}_n \cdot \boldsymbol{k} \equiv s_n - c_n \cdot x_n \mod q \ , \end{cases}$$

where the unknowns are $\boldsymbol{k} = (k_1, \ldots, k_n)$ and $\boldsymbol{a} \cdot \boldsymbol{b}$ denotes the dot product of vectors. The solution $\boldsymbol{k}$ is easily found using linear algebra as soon as vectors $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n$ are linearly independent. Two mutually exclusive cases may occur.

1. $\forall\, i, j \in [1, n]$ with $i \neq j$, one has $\ell_i \neq \ell_j$. Then by Lemma 4, we get

   $$\Pr\left[\det(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_n) = 0\right] = z(n, q) \ .$$

   Then with probability $1 - z(n,q)$, $\mathcal{M}$ recovers $\boldsymbol{k}$ and succeeds in solving n-DL.

2. $\exists\, i, j \in [1, n]$ with $i \neq j$ such that $\ell_i = \ell_j$. Then the reduction $\mathcal{M}$ may fail because it might be the case that $\boldsymbol{\beta}_i = \boldsymbol{\beta}_j$ while $s_i - c_i x_i \not\equiv s_j - c_j x_j \mod q$ resulting in that the system above is not solvable. The probability of this event is unknown and depends on how $\mathcal{R}$ modified its simulation of H between two executions of $\mathcal{A}$. Since distinct executions of $\mathcal{A}$ are not identical and the values of the ℓ_i's are picked pseudo-randomly after all H queries have been made, we invoke Lemma 3 to see that a collision $\ell_i = \ell_j$ occurs with probability

   $$\Pr\left[\exists\, i, j \in [1, n], \ell_i = \ell_j\right] = P(q_H, n) \ .$$

Since $\Pr[\mathcal{M} \text{ fails}] \leq \Pr[\exists\, i, j \in [1, n], \ell_i = \ell_j]$, noting ε'' the success probability of $\mathcal{M}$, we finally get

$$\varepsilon'' \geq \varepsilon' \cdot (1 - P(q_H, n)) \cdot (1 - z(n, q)) \approx \varepsilon' \cdot \exp\left(-\frac{n^2}{2q_H}\right) \cdot \left(1 - \frac{n}{q}\right) \; .$$

The execution time of $\mathcal{M}$ is upper-bounded by $\tau' + \mathsf{poly}\,(\tau', |\mathcal{R}|, n, q_H, \log q)$. $\square$

Our result can be interpreted as follows. When n is smaller than $\sqrt{q_H}$, the ratio $\varepsilon''/\varepsilon'$ remains negligibly close to 1 and the algebraic reduction algorithm $\mathcal{R}$ cannot exist if n-DL is intractable over $\mathbb{G}$. However when $n \gg \sqrt{q_H}$, the ratio $\varepsilon''/\varepsilon'$ becomes rapidly negligibly close to 0 as n increases, allowing $\mathcal{R}$ to exist in the sense that having a substantial ε' does not lead us to solve n-DL with substantial success probability anymore.

8 Conclusion

We believe that our results pose new challenging questions about the standard-model security of common signature schemes. Focusing specifically on Schnorr's scheme, one might wonder what security level is actually reached in real life, as DL cannot be at reach of a *humanly conceivable* reduction. Could Schnorr signatures be proven secure under the CDH or DDH assumption? Can one prove a similar separation with these assumptions? What can be said in this regard about other signature schemes like ElGamal, DSA, GQ, etc. ?

Concerning the random oracle model, we leave it as an open problem to find a more efficient meta-reduction $\mathcal{M}$ that is, to come up with a proof that a factor close to q_H must be lost in any random-oracle-based algebraic reduction $\mathcal{R}$.

9 Acknowledgement

We thank the anonymous reviewers for their valuable feedback. The work described in this document has been financially supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

References

1. ANSI X9.62, *Public-Key fryptography for the financial services industry: the elliptic curve digital standard algorithm (ECDSA)*, American National Standards Institute, 1999.
2. M. Bellare, A. Boldyreva and A. Palacio. *An Un-Instantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem.* Advances in Cryptology - Eurocrypt 2004, LNCS vol. 3027, 2004.
3. M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko, *The One-More-RSA-Inversion Problems and the security of Chaum's Blind Signature Scheme.* J. Cryptology **16** (2003), no. 3, 185–215.

4. M. Bellare and A. Palacio, *GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks.* Advances in Cryptology - Crypto 2002, LNCS vol. 2442, Springer, 2002, pp. 162–177.
5. D. Boneh and R. Venkatesan, *Breaking RSA may not be equivalent to factoring.* Advances in Cryptology - Eurocrypt 1998, LNCS vol. 1233, Springer, 1998, pp. 59–71.
6. E. Brickell, D. Pointcheval, S. Vaudenay and M. Yung, *Design Validations for discrete logarithm based signature schemes.* Publick Key Conference 2000, LNCS vol. 1751, Springer, 2000, pp. 276–292.
7. D. R. L. Brown, *Generic Groups, Collision Resistance and ECDSA*, Des. Codes Cryptography **35** (2005), 119–152.
8. R. Canetti, O. Goldreich, and S. Halevi, *The Random Oracle Methodology, Revisited.* J. Assoc. Comput. Mach. **51** (2004), no. 4, 557–594.
9. FIPS 186. *Digital Signature Standard, Federal Information Processing Standards Publication 186.* US Department of Commerce/NIST, National Technical Information Service, Springfield, Virginia, 1994.
10. A. Dent, *Adapting the weaknesses of the random oracle model to the generic model.* Advances in Cryptology – Asiacrypt 2002, LNCS vol. 2501, Springer, 2002, pp. 100–109.
11. Y. Dodis and L. Reyzin, *On the Power of Claw-Free Permutations.* Third Conference on Security in Communication Networks, SCN 2002, LNCS vol. 2576, Springer, 2003, pp. 55–73.
12. T. El Gamal. *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* IEEE Transactions on Information Theory, vol. IT–31, no. 4, pp. 469–472, 1985.
13. A. Fiat and A. Shamir, *How to Prove Yourself: Practical Solutions to Identification and Signature Problems.* Advances in Cryptology - Crypto 1986, LNCS vol. 263, Springer, 1987, pp. 186–194.
14. S. Goldwasser and Y. Tauman, *On the (In)security of the Fiat-Shamir Paradigm.* FOCS 2003, IEEE Computer Society, 2003, pp. 102–122.
15. L. C. Guillou and J.-J. Quisquater, *A "Paradoxical" Identity-Based Signature Scheme Resulting from Zero-Knowledge.* Advances in Cryptology - Crypto 1988, LNCS vol. 403, Springer, 1990, pp. 216–231.
16. P. Horster, H. Petersen and M. Michels, *Meta-ElGamal signature schemes.* CCS '94: Proceedings of the 2nd ACM Conference on Computer and communications security, ACM Press, 1994, pp. 96–107.
17. KCDSA, *Digital Signature Mechanism with Appendix - Part 2 : Certificate-Based Digital Signature Algorithm (KCDSA)*, TTA.KO -12.0001, 1998.
18. J. B. Nielsen, *Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case.* Advances in Cryptology - Crypto 2002, LNCS vol. 2442, Springer, 2002, pp. 111–126.
19. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures.* J. Cryptology **13** (2000), no. 3, 361–396.
20. C. P. Schnorr, *Efficient signature generation by smart cards.* J. Cryptology **4** (1991), no. 3, 161–174.
21. C. P. Schnorr, *Efficient identification and signatures for smart cards.* Advances in Cryptology - Crypto 1989, LNCS vol. 435, Springer, 1990, pp. 239–251.
22. V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems.* Advances in Cryptology - Eurocrypt 1997, LNCS vol. 1233, Springer, 1997, pp. 256–266.