Design Validations for Discrete Logarithm Based Signature Schemes

Ernest Brickell[1], David Pointcheval[2], Serge Vaudenay[3], and Moti Yung[4]

[1] Intel Inc., Portland, OR, USA
[2] CNRS–LIENS, Paris, France
[3] EPFL, Lausanne, Switzerland
[4] Certco, New York, NY, USA

Abstract. A number of signature schemes and standards have been recently designed, based on the discrete logarithm problem. Examples of standards are the DSA and the KCDSA. Very few formal design/security validations have already been conducted for both the KCDSA and the DSA, but in the "full" so-called random oracle model.
In this paper we try to minimize the use of ideal hash functions for several Discrete Logarithm (DSS-like) signatures (abstracted as generic schemes). Namely, we show that the following holds: "if they can be broken by an existential forgery using an adaptively chosen-message attack then either the discrete logarithm problem can be solved, or some hash function can be distinguished from an ideal one, or multi-collisions can be found." Thus for these signature schemes, either they are equivalent to the discrete logarithm problem or there is an attack that takes advantage of properties of practical hash functions (SHA-1 or whichever high quality cryptographic hash function is used). What is interesting is that the schemes we discuss include KCDSA and slight variations of DSA.
Further, since our schemes are very close to their standard counterparts they benefit from their desired properties (efficiency of computation/space, employment of certain mathematical operations and wide applicability to various algebraic structures). We feel that adding variants with strong validation of security is important to this family of signature schemes since, as we have experienced in the recent past, lack of such validation has led to attacks on standard schemes, years after their introduction. In addition, schemes with formal validation which is made public, may ease global standardization since they neutralize much of the suspicions regarding potential knowledge gaps and unfair advantages gained by the scheme designer's country (e.g. the NSA being the designers of DSS).

1 Introduction

One of the greatest properties of Public-key Cryptography introduced by Diffie and Hellman [9] is the provision of a strong (non-repudiated) integrity function known as "digital signature." The research regarding digital signature schemes has taken a number of basic directions. The first one was theoretical and engaged in reducing the computational assumption required for signature, in order to understand the inherent nature of the primitive. Indeed, digital signature turned out to be equivalent to one-way functions. Another direction has produced various flavors of signatures (blind, undeniable, fail-stop, etc.). The third direction was the design of efficient signature schemes which are very practical and to standardize them.

As part of this third direction, one avenue of research and technology development is the design of digital signature schemes based on the hardness of the discrete logarithm problem (which started with the introduction of the El Gamal signature scheme [10]). A number of efficient schemes have appeared since then and a few of them were standardized, in particular NIST standardized the DSA signature scheme [18].

Whereas the theoretical signature schemes have been presented with a security proof against (existential forgery) attacks [14, 15], the practical schemes were given in an ad-hoc fashion based on intuitive feeling of security. However, as we know in the past and the recent future many schemes believed to be secure have been later broken. Thus the situation is not satisfactory w.r.t. the practical schemes.

When attacking the security of the practical scheme one may attempt a security proof from scratch based solely on computational assumptions. This may not be easily doable since the typical scheme is very specific and it typically employs hashing in addition to involving various operations which were not guided by any structure.

The next proof direction is to assume that certain hash functions which are available and which everyone has a black-box access to, are ideal (i.e., are like a random oracle [2]). Since the available hash functions cannot in fact be random oracles but rather computationally indistinguishable from one, the proof becomes an argument for security: As long as there is no evidence that distinguishes the hash function in use from a random oracle, the security of the scheme is reduced to a well defined number theoretic problem (namely the discrete logarithm). Of course, one has to be careful here. First, only hash functions which are used as a black-box and are replaceable (in case of a specific weakness is found) should be assumed to "look random." Secondly, the methodology does not work universally for every scheme: an artificial theoretical construction was shown which is provably secure under random oracle assumption but becomes insecure under any "concrete implementation" of the oracle [8]. Luckily, this is only an example, and its structure does not apply to the practical signature schemes we study (intuitively, in these schemes there is a separation of the role of the hash function from that of the number theoretic function, such as discrete logarithm).

It is believed that the "random oracle" proof methodology still gives a much better understanding and confidence in a scheme than if a scheme is left completely unanalyzed. In the latter case, any unexpected attack may be mounted, whereas the security study assuming ideal hash, greatly limits the potential attack scenarios.

In this work we study, under the random oracle model, but minimizing the random oracle use, schemes which are, or very closely related to, the standardized schemes (DSA [18], KCDSA [17]). Our goal is to exploit the efficiency of these schemes, yet to modify them slightly if necessary in order to claim validation of security. We believe that the modified schemes are within the spirit of the standards, yet have a strength of being validated. We believe that perhaps the standard bodies should look carefully into our study.

The random oracle methodology was first employed informally by Fiat and Shamir [12], and formalized in Bellare and Rogaway [2]. It was used in showing variants of RSA signatures [24, 4]. At the same time Pointcheval and Stern [21] formalized the Fiat-Shamir technique and then validated security for an El Gamal variant signature, Schnorr signature [25, 26] and Fiat-Shamir signatures. In so doing they formalized the "forking lemma" methodology which we will follow. Further analysis and investigation of multi-signature was performed by Ohta and Okamoto [20]. However, the case of variants of the standardized DSA-like signature which we analyze herein was left open.

Outline of the Paper. In the next Section we present our basic definitions and in Section 3, the basic signature schemes and variants we deal with, namely El Gamal-type signature schemes. In section 4 we present the generic schemes we prove security about, together with some concrete examples. Section 5 presents the basic security result, while Section 6 gives the proof. Example of how to apply the general results to the variants of the standard schemes is given in Section 7, where Section 8 concludes the work.

2 Definitions

In this section, we recall security notions for signature schemes and hash functions together with our computation model, namely the *random oracle model*.

2.1 Security Notions for a Signature Scheme

We first define a signature scheme and then the notions of security. Similar definitions can be found in [15, 21], where the reader is referred for more details.

Definition 1 (A Signature Scheme). *A* signature scheme *consists of three polynomial time randomized algorithms, (Key-Gen, Sign, Ver).*

- *Key-Gen takes as input a random string and outputs a pair of keys (X, Y), where X is the private signature key, and Y is the public verification key.*
- *Sign takes as input a message M and the private signature key X, and produces a signature Sig.*
- *Ver takes as input a message M, a signature Sig, the public verification key Y, and checks whether Sig is a valid signature of M.*

We will use the definition of security of a signature scheme defined by [15], known as existential unforgeability against adaptively chosen-message attacks.

Definition 2 (Unforgeability). *We say that a signature scheme (Key-Gen, Sign, Ver) is* unforgeable *if no adversary who is given the public verification key Y, and the signatures of k messages, $M_1, \ldots, M_k$ adaptively chosen by herself, can produce the signature on a new message M with non-negligible probability.*

2.2 Security Notions for a Hash Function

A hash function is any function which takes as input a message of any length and outputs a digest of fixed size (typically 128 or 160 bits).

Definition 3 (Multi-Collision-Freeness). *A* function *h is said* ℓ-collision-free, *if there is no ℓ-tuple $(x_1, \ldots, x_\ell)$ of pairwise distinct elements such that $h(x_1) = \ldots = h(x_\ell)$.*

But for a hash function, which takes no fixed-length inputs, absence of multi-collisions can not be guaranteed, but may be we can hope the impossibility of finding some of them.

Definition 4 (Multi-Collision-Resistance). *A* function *h is said* ℓ-collision-resistant, *if it is computationally impossible to find an ℓ-tuple $(x_1, \ldots, x_\ell)$ of pairwise distinct elements such that $h(x_1) = \ldots = h(x_\ell)$.*

For simplicity, a *collision-resistant* hash function is, in general, a 2-collision-resistant hash function.

2.3 The Random Oracle Model

In many signature schemes, a cryptographic hash function, such as MD5 [23] or SHA-1 [19], is used, namely to reduce the size of the message. Such a cryptographic hash function has the property that it is collision-resistant, and therefore one-way.

Many recent proofs [3, 4, 20, 21] make the assumption that this cryptographic hash function is an *ideal random function* also known as *random oracle*: for any new query, the answer is uniformly distributed in the output set, independently of previous query/answer pairs. This is the so-called *random oracle model* [2].

Moreover, in this model, a simulator is allowed to set the output of the random oracle to specific values (uniformly distributed) for an input that had not yet been defined. Such a property will be required in the following.

However, since proofs in the random oracle model are just security arguments, but not the strongest proof of security that one could require, we try to minimize the use of random oracles.

3 The El Gamal Type Signature Schemes

El Gamal [10] was the first to propose a signature scheme based on the discrete logarithm problem. Then, Schnorr [25, 26] improved the scheme using the modulo q truncating function, playing in a prime subgroup. This fixes some weaknesses later on discovered by Bleichenbacher [5], van Oorchot and Wiener [27] and also discussed by Anderson and Vaudenay [1]. This latter scheme has been formally proven unforgeable in the random oracle model relative to the discrete logarithm problem [20, 21]. However, as discussed in the introduction many other variants have been defined and standardized by governments: the US-standard DSA [18] and the Korean-standard KCDSA [17]. In both cases, there are system parameters p, q, g such that q and p are primes, $q|p-1$, and g is an element of order q in the group $\mathbb{Z}_p^\star$, *i.e.* the group of invertible integers modulo p. A user has a public key Y, and a private key X such that $Y = g^X \bmod p$.

3.1 The DSA Signature

The Digital Signature Algorithm [18] has been standardized by the US government, together with the hash function SHA-1 [19], denoted by H in the following description. To sign a message M, the Sign algorithm picks a random invertible element k in $\mathbb{Z}_q^\star$ and computes

$$\begin{array}{ll} R = g^k \bmod p & \qquad T = R \bmod q \\ U = H(M) & \qquad S = (U + XT)/k \bmod q \end{array}$$

Formally, the random tape ω defines k and $\mathsf{Sign}(\omega, M) = (S, T)$. The Ver algorithm consists of checking whether

$$\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p\right) \bmod q = T \text{ or not, where } U = H(M).$$

3.2 The KCDSA Signature

The Korean Certificate-based Digital Signature Algorithm [17] has recently been standardized by the Korean government, and is proposed to the IEEE P1363a, as a signature standard. It uses two hash functions G and H. We note that, for efficiency concern, $1/X \bmod q$ is considered as private key instead of X. To sign a message M, the Sign algorithm picks a random element k in $\mathbb{Z}_q^\star$ and computes

$$\begin{array}{ll} R = g^k \bmod p & \qquad T = G(M) \\ U = H(R) & \qquad S = (k - T \oplus U)/X \bmod q. \end{array}$$

Formally, the random tape ω defines k and $\mathsf{Sign}(\omega, M) = (S, U)$. The Ver algorithm consists of first checking the sizes of S and U. Then computing

$$E_G = T \oplus U \qquad W = g^{E_G} Y^S \bmod p$$

and checking whether or not $U = H(W)$, where $T = G(M)$.

3.3 DSA Variants

In order to provide better analyzable schemes than DSA, one can study the following variants, we call DSA–I and DSA–II respectively, which only slightly differ from the original one.

The DSA–I Variant. This first variant differs from the original scheme just by replacing the "$x \mapsto x \bmod q$" truncating function by any hash function G. To sign a message M, the Sign algorithm picks a random invertible element k in $\mathbb{Z}_q^\star$ and computes

$$\begin{array}{ll} R = g^k \bmod p & T = G(R) \\ U = H(M) & S = (U + XT)/k \bmod q. \end{array}$$

Formally, the random tape ω defines k and $\mathsf{Sign}(\omega, M) = (S, T)$. The Ver algorithm consists of checking whether

$$G\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p\right) = T \text{ or not, where } U = H(M).$$

Remark 1. One can easily prove that this DSA–I variant is unforgeable [7, 22] relative to the discrete logarithm problem assuming that both G and H are random oracles. However, this is a very strong assumption which has no real practical impact to the original DSA. Indeed, whereas the SHA-1 function can be assumed practically "ideal", as it is usually done in the random oracle based papers [2–4, 20, 21], the "$x \mapsto x \bmod q$" map hardly can, because of algebraic properties.

The second variant differs a slightly more, but requires weaker assumptions, and namely practical properties (just collision-resistance).

The DSA–II Variant. To sign a message M, the Sign algorithm picks a random invertible element k in $\mathbb{Z}_q^\star$ and computes

$$\begin{array}{ll} R = g^k \bmod p & T = G(R) \\ U = H(M, T) & S = (U + XT)/k \bmod q. \end{array}$$

Formally, the random tape ω defines k and $\mathsf{Sign}(\omega, M) = (S, T)$. The Ver algorithm consists of checking whether

$$G\left(g^{\frac{U}{S}} Y^{\frac{T}{S}} \bmod p\right) = T \text{ or not, where } U = H(M, T).$$

One can note that this second variant has been included in ISO/IEC 14888 [16] as the "Pointcheval-Vaudenay signature".

3.4 Security

As mentioned, KCDSA has already been formal studied [22], however DSA has never been, while they are both serving as national standards!

In the following, we formally define two families of El Gamal-type signature schemes which include the above variants (KCDSA and DSA–II). We then provide security proofs assuming some generic hash functions to be ideal random ones and other just (multi)-collision-resistant/free.

4 The Trusted El Gamal Type Signature Schemes

For the two types of schemes defined in the following, there are

- system parameters p, q, g such that q and p are primes, $q|p-1$, and g is an element of order q in the group $\mathbb{Z}_p^\star$, *i.e.* the group of invertible integers modulo p.
- two hash functions G and H, whose output ranges are denoted by $\mathcal{G}$ and $\mathcal{H}$ respectively. We assume that $q/2 < |\mathcal{G}|, |\mathcal{H}| < 2q$. In both cases, H is considered as an *ideal random function* (or a random oracle), whereas G only requires some practical properties, such as (multi)-collision-resistance or (multi)-collision-freeness.
- three functions:

  $$F_1 : (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q \quad F_2 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q \quad F_3 : (\mathbb{Z}_q, \mathcal{G}, \mathcal{H}) \longrightarrow \mathbb{Z}_q$$

  satisfying for all $(a, b, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{G}, \mathcal{H})$,

  $$F_2(F_1(a, b, T, U), T, U) + b \cdot F_3(F_1(a, b, T, U), T, U) = a \bmod q.$$

and a user has private and public keys X, Y, such that $Y = g^X \bmod p$.

Definition 5 (The TEGTSS Verification Equation). *A tuple* (W, S, T, U) *is said to satisfy the* TEGTSS verification equation *if for* $E_G = F_2(S, T, U)$*, and* $E_Y = F_3(S, T, U)$ *then* $W = g^{E_G} Y^{E_Y}$ *mod* p*.*

Then, Trusted El Gamal Types Signature Schemes are of two distinct types, depending on the use of the functions G and H.

4.1 Type I: the TEGTSS–I schemes.

- To sign a message M, the signer chooses an element k at random in $\mathbb{Z}_q^\star$, computes $T = G(M)$ and generates $R = g^k \bmod p$. He then gets $U = H(R)$ and computes $S = F_1(k, X, T, U)$.
  The signature of M is the triple (S, T, U). In practice, the pair (S, U) is enough since $T = G(M)$, but we keep the triple for the reader's convenience.
- To verify the signature (S, T, U) of the message M, a verifier computes $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$ and finally $W = g^{E_G} Y^{E_Y} \bmod p$. He then checks whether $T = G(M)$ and $U = H(W)$ or not.

TEGTSS–I Properties. To provide a TEGTSS–I scheme, F_3 must satisfy the following conditions for tuples (W, S_i, T_i, U_i) for $i = 1, 2$ that satisfy the *TEGTSS Verification Equation*:

1. if $T_1 \neq T_2$, then $F_3(S_1, T_1, U_1) \neq F_3(S_2, T_2, U_2)$.
2. For a fixed verifying tuple (W, S_1, T_1, U_1) there is a one-to-one map between the values of U_2 and the values of T_2 such that (W, S_2, T_2, U_2) verifies the TEGTSS equation and $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$.

Example: the KCDSA Signature. Both signature and verification algorithms are exactly as described above, with the following functions:

$$F_1(k, X, T, U) = (k - T \oplus U)/X \bmod q$$

$$F_2(S, T, U) = T \oplus U \bmod q \text{ and } F_3(S, T, U) = S \bmod q,$$

where $T = G(M)$, $R = g^k \bmod p$, $U = H(R)$ and $S = F_1(k, X, T, U)$.

Lemma 1. *The KCDSA signature is a TEGTSS–I scheme.*

Proof. We need to show that the functions F_1, F_2 and F_3 satisfy the properties of a TEGTSS–I scheme:

– for all $(k, X, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{Q}, \mathcal{H})$,

$$F_2(F_1(k, X, T, U), T, U) + X \times F_3(F_1(k, X, T, U), T, U)$$
$$= T \oplus U + X \times (k - T \oplus U)/X = k \bmod q,$$

– if $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$, then $S_1 = S_2 = S$ and since W is fixed, $F_2(S, T_1, U_1) = F_2(S, T_2, U_2)$ and $U_1 = U_2$. Therefore $T_1 \oplus U_1 = T_2 \oplus U_2$ and consequently $T_1 = T_2$.

– suppose $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$ for a given W. Then $S_1 = S_2 = S$. Since W is fixed, $F_2(S, T_1, U_1) = F_2(S, T_2, U_2)$: $T_1 \oplus U_1 = T_2 \oplus U_2$. Therefore $T_2 = T_1 \oplus U_1 \oplus U_2$.

□

4.2 Type II: the TEGTSS–II schemes.

– To sign a message M, the signer chooses an element k at random in $\mathbb{Z}_q^\star$, computes $R = g^k \bmod p$ and $T = G(R)$. He then gets $U = H(M, T)$ and computes $S = F_1(k, X, T, U)$.
  The signature of M is the triple (S, T, U). In practice, the pair (S, T) is enough, since $U = H(M, T)$. But the triple is kept for the reader's convenience.

– To verify the signature (S, T, U) on the message M, a verifier computes $E_G = F_2(S, T, U)$ and $E_Y = F_3(S, T, U)$ and finally $W = g^{E_G} Y^{E_Y} \bmod p$. He then checks whether $T = G(W)$ and $U = H(M, T)$ or not.

TEGTSS–II Properties. To provide a TEGTSS–II scheme, the functions F_2 and F_3 must satisfy the following one-to-one condition: for given T, E_G and E_Y, there exists a unique pair (U, S) such that

$$E_G = F_2(S, T, U) \text{ and } E_Y = F_3(S, T, U).$$

Furthermore, this pair is easy to find.

Example: the DSA–II Variant. This DSA variant is exactly as described above, with the following functions:

$$F_1(k, X, T, U) = (U + XT)/k \bmod q$$
$$F_2(S, T, U) = U/S \bmod q \text{ and } F_3(S, T, U) = T/S \bmod q,$$

where $R = g^k \bmod p$, $T = G(R)$, $U = H(M, T)$ and $S = F_1(k, X, T, U)$.

Lemma 2. *The DSA–II signature is a TEGTSS–II scheme.*

Proof. We need to show that the functions F_1, F_2 and F_3 satisfy TEGTSS–II properties:

– for all $(k, X, T, U) \in (\mathbb{Z}_q, \mathbb{Z}_q, \mathcal{Q}, \mathcal{H})$,
$$F_2(F_1(k, X, T, U), T, U) + X \times F_3(F_1(k, X, T, U), T, U)$$
$$= U/S + XT/S = (U + XT)/S = k \bmod q,$$
– for given T, E_G and E_Y, $F_2(S, T, U) = E_G$ and $F_3(S, T, U) = E_Y$ imply $S = T/E_Y \bmod q$ and $U = SE_G \bmod q$.

□

5 Security Results

In this paper, we claim the following security results for Trusted El Gamal Type Signature Schemes of both types.

Theorem 1. *Let us consider an attacker $\mathcal{A}$ against a Trusted El Gamal Type Signature Scheme. Let us assume that $\mathcal{A}$ is able to perform an existential forgery under an adaptively chosen-message attack with probability $\varepsilon > 4/q$ after Q queries to the H function.*

– *for Type I schemes, if G is collision-resistant and H a random oracle, then one extracts the secret key X with less than $25Q/\varepsilon$ replays of $\mathcal{A}$, with constant probability greater than $1/100$.*
– *for Type II schemes,*
  - *if G is $(\ell + 1)$-collision-resistant*
  - *if $x \mapsto G(g^x \bmod p)$ is $(\ell + 1)$-collision-free*

  and H a random oracle, then one extracts the secret key X with less than $25Q\ell \log(2\ell)/\varepsilon$ replays of $\mathcal{A}$ (where $\log$ denotes the logarithm is basis 2), with constant probability greater than $1/100$.

6 Proof of the Security Results

6.1 General Method of Proof

We construct a simulator that produces signatures of a given message in a reasonable (poly) time in a way indistinguishable from the signer's. Therefore, if the attacker could construct a successful adaptively chosen-message attack using the legitimate signer, then she would be able to do so using only the simulator. Then we use a forking lemma as in [21] to show that if the attacker can construct a signature with a specific ideal hash function, she can (with non-negligible probability) construct many signatures with the same fixed values, but in which the ideal hash functions output different answers. We then show that two such signatures can be used to compute the discrete logarithm of the public key, thus solving the discrete logarithm problem.

In their papers [21], Pointcheval and Stern defined particular subcases of the Type II signatures, where G is the identity and therefore clearly multi-collision-resistant and even collision-free. Here, we need similar tools, namely their "splitting lemma" and an improved version of their "forking lemma". The "splitting lemma" is a formal probabilistic version of the "heavy rows lemma" [11, 20].

Lemma 3 (The Splitting Lemma). *Let $A \subset X \times Y$ and we assume that $\Pr[(x,y) \in A] \geq \varepsilon$. Define*

$$B = \left\{ (x,y) \in X \times Y \;\middle|\; \Pr_{y' \in Y}[(x,y') \in A] \geq \frac{\varepsilon}{2} \right\} \quad \textit{and} \quad \bar{B} = (X \times Y) \backslash B,$$

then the following statements hold:

(i) $\Pr[B] \geq \varepsilon/2$
(ii) $\forall (x,y) \in B, \Pr_{y' \in Y}[(x,y') \in A] \geq \varepsilon/2$.
(iii) $\Pr[B \mid A] \geq 1/2$.

Proof. See Pointcheval–Stern's [21] or Ohta-Okamoto's [20] papers. □

For any $\ell \leq \sqrt{q}/4$, one can state the following variant of the forking lemma [21].

Lemma 4 (The Improved Forking Lemma). *Let us consider a probabilistic polynomial time Turing machine $\mathcal{A}$, called the attacker, and a probabilistic polynomial time simulator $\mathcal{B}$. If $\mathcal{A}$ can find with probability $\varepsilon > 4/q$ a verifiable tuple (M, R, S, T, U) with less than Q queries to the hash function, for a new message M and for a U directly defined by H, then with a constant probability $1/96$, with $(1+24Q\ell\log(2\ell))/\varepsilon$ replays of $\mathcal{A}$ and $\mathcal{B}$ with different random oracles, $\mathcal{A}$ will output $\ell+1$ verifiable tuples $(M_i, R_i, S_i, T_i, U_i)_{i=1,\ldots,\ell+1}$ such that the U_i are pairwise distinct, and all the R_i equal for TEGTSS–I schemes but all the (M_i, T_i) equal for TEGTSS–II schemes.*

Proof. Let Ω and Φ denote the sets of all random tapes that could be used by the attacker $\mathcal{A}$ and the simulator $\mathcal{B}$ respectively, and let ω denote an arbitrary random tape in Ω, and let ϕ denote an arbitrary random tape in Φ. Let Ψ denote the set of all random tapes that define the random oracle H and let ψ denote an arbitrary random tape in Ψ. The attacker $\mathcal{A}$ can query H directly by requesting the value of $H(X)$ for some X or $\mathcal{A}$ can cause a query of H indirectly by asking $\mathcal{B}$ for a signature of a message M. During the execution of the protocol, let $\mathcal{Q}_1, \ldots, \mathcal{Q}_Q$, denote the ordered set of direct queries for H. We assume that $U = H(\mathcal{Q}_j)$, with $\mathcal{Q}_j = R$ for TEGTSS–I schemes, and $\mathcal{Q}_j = (M, T)$ for TEGTSS–II schemes, for some $j \leq Q$, since $\mathcal{A}$ would have to know the random answer U to be able to determine if (M, R, S, T, U) was a verifiable tuple, excepted with probability $\nu \leq 2/q \leq \varepsilon/2$.

Therefore, the probability over the choice of ω, ϕ and ψ such that $\mathcal{A}$ outputs a new verifiable tuple, (M, R, S, T, U), after Q values, for a new message M and U directly defined by H (and not by the simulator $\mathcal{B}$) is at least $\varepsilon - \nu \geq \varepsilon/2$. We say that (ω, ϕ, j, ψ) is a winning input if for tapes (ω, ϕ, ψ), $\mathcal{A}$ outputs a verifiable tuple (M, R, S, T, U) after Q queries, for a new message M, in which $\mathcal{Q}_j = R$ or $\mathcal{Q}_j = (M, T)$, respectively: $\mathcal{Q}_j$ is the crucial query.

By the Splitting-Lemma (Lemma 3), there exists a set $\Gamma \subseteq \Omega \times \Phi$ such that $\Pr[(\omega, \phi) \in \Gamma] \geq \varepsilon/4$, and for $(\omega, \phi) \in \Gamma$, $\Pr[\exists j, (\omega, \phi, j, \psi) \text{ winning}] \geq \varepsilon/4$. Furthermore,

$$\Pr[(\omega, \phi) \in \Gamma \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

For each $(\omega, \phi) \in \Gamma$, let us define $\mathcal{J}(\omega, \phi)$ to be the set of indices $j \leq Q$ such that $\Pr[(\omega, \phi, j, \psi) \text{ winning}] \geq \varepsilon/8Q$. Since the number of possible j is upper-bounded by Q, one can easily prove by contradiction that $\mathcal{J}(\omega, \phi) \neq \emptyset$ and

$$\Pr[j \in \mathcal{J}(\omega, \phi) \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

For $(\omega, \phi) \in \Gamma$ and $j \in \mathcal{J}(\omega, \phi)$, define a partition of the set Ψ into Hash-Classes, where a Hash-Class is defined by a tuple $(P_1, P_2, \ldots, P_{j-1})$ and $\psi \in \Psi$ is in the Hash-Class $(P_1, P_2, \ldots, P_{j-1})$ if $H(\mathcal{Q}_i) = P_i$ for $i \leq j-1$ for all queries $\mathcal{Q}_i$ for $i \leq j-1$ that result from running (ω, ϕ, ψ). Moreover, $\Psi_{\omega,\phi,j,\psi}$ will be defined to be the Hash-Class $(P_1, P_2, \ldots, P_{j-1})$ where $H(\mathcal{Q}_i) = P_i$ for $i \leq j-1$ for all queries $\mathcal{Q}_i$ for $i \leq j-1$ that result from running (ω, ϕ, ψ).

By the Splitting-Lemma (Lemma 3), there exists a set of Hash-Classes, $\Theta(\omega, \phi, j)$ such that $\Pr[\psi \in \Theta(\omega, \phi, j)] \geq \varepsilon/16Q$ and for each $\psi \in \Theta(\omega, \phi, j)$, $\Pr[(\omega, \phi, j, \psi') \text{ winning} \mid \psi' \in \Psi_{\omega,\phi,j,\psi}] \geq \varepsilon/16Q$. Furthermore,

$$\Pr[\psi \in \Theta(\omega, \phi, j) \mid (\omega, \phi, j, \psi) \text{ winning}] \geq \frac{1}{2}.$$

To generate two verifiable tuples, tapes ω, ϕ are chosen at random. If $\mathcal{A}$ does not forge a signature, then new tapes ω, ϕ are chosen, until a forgery $(M_1, R_1, S_1, T_1, U_1)$ occurs, with $\mathcal{Q}_j$ as crucial query and Hash-Class $\Psi_{\omega,\phi,j,\psi}$. This is a "good" forgery if $(\omega, \phi) \in \Gamma$, $j \in \mathcal{J}(\omega, \phi)$ and $\psi \in \Theta(\omega, \phi, j)$, which happens with probability greater than

$$\begin{aligned}
&\Pr[\psi \in \Theta(\omega, \phi, j) \mid (\omega, \phi, j, \psi) \text{ winning} \wedge\ (\omega, \phi) \in \Gamma \wedge\ j \in \mathcal{J}(\omega, \phi)] \\
&\times \Pr[j \in \mathcal{J}(\omega, \phi) \mid (\omega, \phi, j, \psi) \text{ winning} \wedge\ (\omega, \phi) \in \Gamma] \\
&\times \Pr[(\omega, \phi) \in \Gamma) \mid (\omega, \phi, j, \psi) \text{ winning}]\ \geq \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8}.
\end{aligned}$$

Thereafter, one fixes tapes ω, ϕ and chooses ψ' at random in Hash-Class $\Psi_{\omega,\phi,j,\psi}$. $\mathcal{A}$ is run again for $24Q \log(2\ell)/\varepsilon$ repetitions (where log denotes the logarithm in basis 2) or until another forgery is produced with $\mathcal{Q}_j$ as crucial query: such an event occurs with probability greater than $(1 - 1/2\ell)$, if the first forgery was a "good" one. One repeats this process $\ell - 1$ other times, and therefore gets ℓ more forgeries with probability greater than $(1 - 1/2\ell)^\ell \geq 1/3$.

With probability greater than $\left(1-\ell/|\mathcal{H}|\right)^{\ell} \geq e^{-4\ell^2/q}$, the U_i are pairwise distinct. Finally, one gets the following probability of success, if $\ell \leq \sqrt{q}/4$,

$$
\begin{aligned}
&\Pr\left[\begin{array}{c}\ell+1 \text{ winning inputs after } (1+24Q\ell\log(2\ell))/\varepsilon \text{ trials} \\ \text{with pairwise distinct } U_i\end{array}\right] \\
&\geq \Pr[\text{a winning input after } 1/\varepsilon \text{ trials}] \\
&\quad \times \Pr[\text{a “good” winning input} \mid \text{ winning input}] \\
&\quad \times \Pr\left[\begin{array}{c}\ell \text{ other winning inputs} \\ \text{after } 24Q\ell\log(2\ell)/\varepsilon \text{ trials}\end{array} \,\middle|\, \text{“good” winning input}\right] \times e^{-4\ell^2/q} \\
&\geq \frac{1}{3}\times\frac{1}{8}\times\frac{1}{3}\times\frac{3}{4} = \frac{1}{96} \geq \frac{1}{100}.
\end{aligned}
$$

□

In the following, we apply this forking lemma to prove the security of both families, which include KCDSA and DSA–II.

6.2 TEGTSS – Type I

Let us start with the Type I (which includes the KCDSA scheme), proving first the existence of an indistinguishable simulator.

Lemma 5. *Suppose H is an ideal random function with output between 0 and $|\mathcal{H}|-1$. Then there exists a simulator that creates verifiable tuples such that after b steps, the probability that the simulator can be distinguished from a signer is less than $b^2/2q$.*

Proof. Given a message M to be signed, the simulator generates tuples by computing $T = G(M)$, then picking U at random between 0 and $|\mathcal{H}|-1$ and S at random between 0 and $q-1$. The simulator computes $E_G = F_2(S,T,U)$, $E_Y = F_3(S,T,U)$ and $R = g^{E_G}Y^{E_Y} \bmod p$. It then defines $H(R)$ to be equal to U. In the event that $H(R)$ was already defined, the simulator would declare failure. These tuples will be uniformly distributed among all verifiable tuples $(R,S,T=G(M),U)$ such that $R = g^{E_G}Y^{E_Y} \bmod p$.

The adversary can only distinguish between this distribution and the signer's one if the simulator computes an R for which $H(R)$ was already defined or if the signer computes an R which he had used earlier. Let b denote the number of queries that have been made to the random oracle H. The probability of one of these events happening is less than $1-e^{-b(b-1)/2q}$ (birthday paradox) which can be approximated by $b^2/2q$. □

Theorem 2. *Suppose that H is an ideal random function but G a collision-resistant hash function. Given an attacker $\mathcal{A}$ that can find with probability ε a verifiable tuple (M,R,S,T,U) for a new message M, with less than Q queries to the hash function H, then with constant probability $1/96$, with less than $25Q/\varepsilon$ replays of $\mathcal{A}$, with different random oracles, $\mathcal{A}$ extracts the secret key X.*

Proof. When the attacker $\mathcal{A}$ outputs a new verifiable tuple (M, R, S, T, U), either $H(R)$ had been defined by the simulator (case 1) or directly by H (case 2).

- case 1: the simulator had produced a verifiable tuple, (M', R, S', T', U'), for which $M \neq M'$ and therefore $T = G(M) \neq G(M') = T'$, since G is collision-resistant. Because of the TEGTSS–I properties, one has two distinct representations of the same R in the basis (g, Y), which leads to X [6].
- case 2: $\mathcal{A}$ outputs a verifiable tuple, (M, R, S, T, U), in which $R = \mathcal{Q}_j$ for some $j \leq Q$ and $\mathcal{A}$ made a direct query for the value of $H(\mathcal{Q}_j)$. Using the forking lemma (Lemma 4), after less than $(1+24Q)/\varepsilon$ replays of $\mathcal{A}$, one gets two tuples (M_1, R, S_1, T_1, U_1) and (M_2, R, S_2, T_2, U_2) such that $U_1 \neq U_2$. With a closer look to the proof of the forking lemma, one can see that U_2 follows the uniform distribution. Given U_2, let T_2 be the unique value such that $F_3(S_1, T_1, U_1) = F_3(S_2, T_2, U_2)$ for a verifiable tuple. By the assumption that G is collision-resistant, and therefore one-way, the probability that $\mathcal{A}$ can find a message M_2 such that $G(M_2) = T_2$ is vanishingly small. Consequently, we likely have $F_3(S_1, T_1, U_1) \neq F_3(S_2, T_2, U_2)$ and thus two distinct representations of the same R in the basis (g, Y), which leads to X [6].

□

6.3 TEGTSS – Type II

Let us now study the Type II (which includes the DSA–II scheme), proving first the existence of an indistinguishable simulator.

Lemma 6. *Suppose H is an ideal random function with output between 0 and $|\mathcal{H}|-1$. Then there exists a simulator that creates verifiable tuples such that after b steps, the probability that the simulator can be distinguished from a signer is less than $b^2/2q$.*

Proof. Given a message M to be signed, the simulator generates tuples by first picking A and B at random, both in $\mathbb{Z}_q$. It then computes $R = g^A Y^B \bmod p$ and $T = G(R)$. Using the property of F_2 and F_3, S and U are defined as the only values leading to both $E_G = A$ and $E_Y = B$. Then $H(M, T)$ is defined to be equal to U. As above, this simulation is indistinguishable but with an advantage upper-bounded by $b^2/2q$. □

Theorem 3. *Let us assume that H is an ideal random function and G an $(\ell+1)$-collision-resistant function. Given an attacker $\mathcal{A}$ that can find with probability ε a verifiable tuple (M, R, S, T, U) for a new message M, with less than Q queries to the hash function H, then with constant probability $1/96$, with less than $25Q\ell\log(2\ell)/\varepsilon$ replays of $\mathcal{A}$, with different random oracles, $\mathcal{A}$ extracts the secret key X.*

Proof. $\mathcal{A}$ outputs a verifiable tuple, (M, R, S, T, U), in which $(M, T) = \mathcal{Q}_j$ for some $j \leq Q$ and $\mathcal{A}$ made a direct query for the value of $H(\mathcal{Q}_j)$, since M is a new message, never asked to the simulator. Using the forking lemma (Lemma 4), after less than $(1+24Q\ell\log(2\ell))/\varepsilon$ replays of $\mathcal{A}$, one gets $\ell+1$ tuples (M, R_i, S_i, T, U_i) such that the U_i are pairwise distinct, with $T = G(R_i)$.

Since G is $(\ell+1)$-collision-resistant, there exists a pair of indices (i, j) for which we have $R_i = R_j$. Assume that $E_{G\,i} = E_{G\,j}$ and $E_{Y\,i} = E_{Y\,j}$. Then, because of the TEGTSS–II properties, $S_i = S_j$ and $U_i = U_j$, which contradicts the fact that the U_i are all distinct. Then from two representations of the same R in basis (g, Y), one gets X [6]. ◻

Theorem 4. *Let us assume that H is an ideal random function and $x \mapsto G(g^x \bmod p)$ an $(\ell+1)$-collision-free function. Given an attacker $\mathcal{A}$ that can find with probability ε a verifiable tuple (M, R, S, T, U) for a new message M, with less than Q queries to the hash function H, then with constant probability $1/96$, with less than $25Q\ell\log(2\ell)/\varepsilon$ replays of $\mathcal{A}$, with different random oracles, $\mathcal{A}$ extracts the secret key X.*

Proof. As above, after less than $(1+24Q\ell\log(2\ell))/\varepsilon$ replays of $\mathcal{A}$, one gets $\ell+1$ tuples (M, R_i, S_i, T, U_i) such that the U_i are pairwise distinct, with $T = G(R_i)$ and $R_i = g^{E_{G\,i}} Y^{E_{Y\,i}} = g^{x_i} \bmod p$ for some x_i. Since $x \mapsto G(g^x \bmod p)$ is $(\ell+1)$-collision-free, the same conclusion as above holds. ◻

7 Application to some Signature Schemes

Lemma 7. *If G is just collision-resistant but H a random oracle, the KCDSA is unforgeable relative to the discrete logarithm problem.*

Proof. It is an immediate corollary from Lemma 1 and Theorem 1. ◻

7.1 The DSA–II Variant

Lemma 8. *If G is an $(\ell+1)$-collision-resistant function or $x \mapsto G(g^x \bmod p)$ is an $(\ell+1)$-multi-collision-free function, but H a random oracle, then DSA–II is unforgeable relative to the discrete logarithm problem.*

Proof. It is an immediate corollary from Lemma 2 and Theorem 1. ◻

Remark 2. One can first remark that for any random function G, the probability that $x \mapsto G(g^x \bmod p)$ has a $(\ell+1)$-multi-collision is approximately less than $eq/(\ell+1)!$ [22], which is very small for $\ell = \log q$.

This provides a *security argument* for a very slight variant of the original DSA, where the $H(M)$ is just replaced by $H(M, T)$. Indeed, it is very unlikely that the "$x \mapsto (g^x \bmod p) \bmod q$" map has $(\log q)$-multi-collision. Indeed, even just a 2-collision would lead to an important weakness in the original DSA design by an attack similar than Vaudenay's [28]: if for a given (p, q, g) provided by a honest authority someone happens to find out a 2-collision

$$T = g^k \bmod p \bmod q = g^{k'} \bmod p \bmod q$$

then he can choose two different messages M and M' and a particular X as his private key so that the signatures of M and M' collide. Namely, if

$$X = \frac{kH(M') - k'H(M)}{T(k - k')} \bmod q$$

then $(H(M) + XT)/k \bmod q = S = (H(M') + XT)/k' \bmod q$ so that he can reveal the signature (S, T) of M and later on claim it was the signature of M'.

7.2 Short-Length Signatures

The Type II of TEGTSS has the attractive property to provide short signatures. Indeed, a hash function that returns 80-bits digests can be considered as 5-collision-resistant, since a research would require 2^{64} computations [13]. Therefore, since the practical signature consists of the pair (S, T) where $S \in \mathbb{Z}_q$ and T the digest produced by G, a 5-collision-resistant hash function, it can be shorter than 30 bytes-long.

8 Conclusion

We have studied and validated security (under the random oracle model) of schemes which are, or very closely related to, some standardized schemes (DSA, KCDSA). We proved their security while maintaining the efficiency of the standard schemes. We, therefore, believe that the standard bodies should look carefully into our study.

References

1. R. Anderson and S. Vaudenay. Minding your p's and q's. In *Asiacrypt '96*, LNCS 1163, pages 26–35. Springer-Verlag, Berlin, 1996.
2. M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCCS*, pages 62–73. ACM Press, New York, 1993.
3. M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.
4. M. Bellare and P. Rogaway. The Exact Security of Digital Signatures – How to Sign with RSA and Rabin. In *Eurocrypt '96*, LNCS 1070, pages 399–416. Springer-Verlag, Berlin, 1996.
5. D. Bleichenbacher. Generating El Gamal Signatures without Knowing the Secret Key. In *Eurocrypt '96*, LNCS 1070, pages 10–18. Springer-Verlag, Berlin, 1996.
6. S. A. Brands. An Efficient Off-Line Electronic Cash System Based on the Representation Problem. Technical Report CS-R9323, CWI, Amsterdam, 1993.
7. E. F. Brickell. Invited lecture given at the Crypto '96 conference. Unpublished.
8. R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, New York, 1998.

9. W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.
10. T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, IT–31(4):469–472, July 1985.
11. U. Feige, A. Fiat, and A. Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology*, 1:77–95, 1988.
12. A. Fiat and A. Shamir. How to Prove Yourself: Practical Solutions of Identification and Signature Problems. In *Crypto '86*, LNCS 263, pages 186–194. Springer-Verlag, Berlin, 1987.
13. M. Girault and J. Stern. On the Length of Cryptographic Hash-Values used in Identification Schemes. In *Crypto '94*, LNCS 839, pages 202–215. Springer-Verlag, Berlin, 1994.
14. S. Goldwasser, S. Micali, and R. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984.
15. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.
16. ISO. ISO/IEC 14888 Final Draft – Information Technology – Security Techniques - Digital Signatures with Appendix. International Organization for Standardization, Berlin, Germany, 1998.
17. KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. IEEE P1363a Submission, August 1998.
Available from `http://grouper.ieee.org/groups/1363/addendum.html`.
18. NIST. *Digital Signature Standard* (DSS). Federal Information Processing Standards PUBlication 186, November 1994.
19. NIST. *Secure Hash Standard* (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.
20. K. Ohta and T. Okamoto. On Concrete Security Treatment of Signatures Derived from Identification. In *Crypto '98*, LNCS 1462, pages 354–369. Springer-Verlag, Berlin, 1998.
21. D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 1999.
Available from `http://www.di.ens.fr/~pointche`.
22. D. Pointcheval and S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical Report LIENS-96-17, LIENS, October 1996.
23. R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, The Internet Engineering Task Force, April 1992.
24. R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.
25. C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251. Springer-Verlag, Berlin, 1990.
26. C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.
27. P. C. van Oorschot and M. J. Wiener. On Diffie-Hellman Key Agreement with Short Exponents. In *Eurocrypt '96*, LNCS 1070, pages 332–343. Springer-Verlag, Berlin, 1996.
28. S. Vaudenay. Hidden Collisions on DSS. In *Crypto '96*, LNCS 1109, pages 83–88. Springer-Verlag, Berlin, 1996.