On the One-Per-Message Unforgeability of (EC)DSA and Its Variants

Manuel Fersch[1(✉)], Eike Kiltz[1], and Bertram Poettering[1,2]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Bochum, Germany
{manuel.fersch,eike.kiltz}@rub.de
[2] Information Security Group, Royal Holloway, University of London, London, UK
bertram.poettering@rhul.ac.uk

Abstract. The American signature standards DSA and ECDSA, as well as their Russian and Chinese counterparts GOST 34.10 and SM2, are of utmost importance in the current security landscape. The mentioned schemes are all rooted in the Elgamal signature scheme (1984) and use a hash function and a cyclic group as building blocks. Unfortunately, authoritative security guarantees for the schemes are still due: All existing positive results on their security use aggressive idealization approaches, like the generic group model, leading to debatable overall results.

In this work we conduct security analyses for a set of classic signature schemes, including the ones mentioned above, providing positive results in the following sense: If the hash function (which is instantiated with SHA1 or SHA2 in a typical DSA/ECDSA setup) is modeled as a random oracle, and the signer issues at most one signature per message, then the schemes are unforgeable if and only if they are key-only unforgeable, where the latter security notion captures that the adversary has access to the verification key but not to sample signatures. Put differently, for the named signature schemes, in the one-signature-per-message setting the signature oracle is redundant.

Keywords: Elgamal signatures · DSA · ECDSA · GOST · SM2

1 Introduction

Digital signatures. Digital signature schemes are a ubiquitous cryptographic primitive. They are extensively used for message and entity authentication and find widespread application in real-world protocols. The signature schemes most often used in practice are likely the RSA-based PKCS#1v1.5, and the DLP-based DSA and ECDSA [20]. For instance, current versions of TLS exclusively employ signatures of these types to authenticate servers. Standardized schemes

The full version of this work can be found on the *IACR Cryptology ePrint Archive* [12].

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part II, LNCS 10678, pp. 519–534, 2017.
https://doi.org/10.1007/978-3-319-70503-3_17

that share a great similarity with (EC)DSA are the Russian GOST 34.10 [9] and the Chinese SM2 [19]. In the following we describe those schemes in more detail.

DSA AND ECDSA. The signature schemes DSA and ECDSA build on ideas of ElGamal [10] and are defined over a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. They utilize two independent hash functions, H and f, that map messages and group elements, respectively, into the exponent space $\mathbb{Z}_q$. Function f is called the *conversion function*. While for DSA the group $\mathbb{G}$ is a prime-order subgroup of the multiplicative group of some prime field $\mathrm{GF}(p)$ with the canonical representation of group elements as integers in $\{1, \ldots, p-1\}$, and f is defined as $A \mapsto (A \bmod p) \bmod q$, for ECDSA the group is a subgroup of an elliptic curve over some field $\mathrm{GF}(p^n)$, and f is defined as $A \mapsto A.x \bmod q$ where $A.x$ is an encoding of the x-coordinate of elliptic curve point A as an integer.

The signature schemes GOST and SM2 use similar settings. After having fixed the cyclic group $\mathbb{G}$, the hash function H, and the conversion function f, if x is a signing key and $X = g^x$ the corresponding verification key, an (EC)DSA signature on a message m is a pair (s, t) such that $s = (H(m) + xt)/r$ and $t = f(g^r)$, where r is freshly picked in each signing operation. In GOST and SM2, different equations that values s, t, r, x have to fulfill are used. (For details see Fig. 2.)

PRIOR ANALYSES OF ELGAMAL-TYPE SIGNATURE SCHEMES. The first positive results on (unmodified) ECDSA are due to Brown. In [4–6] he proves security of ECDSA in the *generic group model* [27]. Unfortunately, some crucial formal aspects of his idealization remain unclear, for instance that his modeling approach for the group *implicitly* also idealizes the conversion function f. This has unexpected impact: he *de facto* proves that ECDSA signatures are strongly unforgeable, while in practice this is obviously not the case. See the discussions in [11,28] for more details. Further, as Brown reports, his arguments are applicable to ECDSA only, but not to the (closely-related) DSA.

Independently of the findings discussed above, in [4,6,7] Brown identifies both sufficient and necessary conditions on H, f for the security of ECDSA. However, the sufficient ones are significantly stronger than the discrete logarithm problem.

In an informal discussion, in [6, II.4.4], Brown mentions that for ECDSA, in the random oracle model, unforgeability against adversaries that have access to the verification key but not to a signing oracle implies unforgeability against adversaries that can request signatures, but at most one per message. No formal argument is given for this claim. We work out the details in the current article. As our treatment shows, a formal proof requires careful consideration and additional techniques.

In [11] the current authors propose GenDSA, a signature framework that subsumes both DSA and ECDSA in unmodified form, and prove the unforgeability of corresponding signatures using a novel approach of idealization: They decompose the conversion function into three independent functions, where the outer two mimic algebraic properties of the conversion function's domain and

range, and the inner function is modeled as a bijective random oracle.[1] In the full version they extend their results to also cover GOST and SM2. To the best of our knowledge, this is the only existing security proof for GOST signatures. For SM2, the only other security evaluation is in the generic group model [31].

In comparison to [5,11] the current work takes a conservative approach: We idealize neither the group nor the conversion function but rather model a hash function as a random oracle. As this hash function is typically instantiated with a dedicated construction like SHA1 or SHA2, we believe our assumptions are weaker and thus preferable to those used in [5,11,31]. We caution, however, that also our results are weaker for not giving a reduction to the DLP, but to a different (non-interactive) assumption.

Further related work. The works discussed next do not establish security results for standardized schemes like DSA/GOST/SM2: Some works instead target modified versions of these schemes, others give implementation advice.

Brickell et al. [3] define a framework for signature schemes called *Trusted El Gamal Type Signature Scheme* and prove its unforgeability in the random oracle model. Among the instantiations of their framework are the schemes DSA-I (reportedly due to Brickell, 1996) in which the conversion function f is replaced by a random oracle, and DSA-II (due to [26]) that deviates from DSA for applying the hash function H to both the message and the ephemeral value $f(g^r)$. The framework of [3] cannot be instantiated such that unmodified (EC)DSA, GOST, or SM2 is covered.

Similarly, Malone-Lee and Smart [22] propose the variants ECDSA-II and ECDSA-III of ECDSA. In order to make certain attacks impossible (like duplicate signatures [28] where one signature is valid for two messages), and for obtaining tighter security reductions, the authors diverge from the original ECDSA scheme.

Other work on the security of DSA and ECDSA, identifying necessary conditions for the security of the schemes or analyzing their robustness against flaws in implementations and parameter selection, was conducted by Vaudenay [29,30], Howgrave-Graham and Smart [18], Nguyen and Shparlinski [24], Leadbitter et al. [21], García et al. [13], and Genkin et al. [14].

Our Contribution

Our contribution is threefold. First, we describe the abstract signature scheme GenElgamal that, among others, subsumes DSA, ECDSA, and GOST in unmodified, and SM2 in an equivalent form. Second, we show that in the random oracle model (for H), forging signatures in the presence of a signing oracle that can be queried at most once on each message (one-per-message unforgeability, uf-cma1) is as hard, but with a non-tight security reduction, as without such an oracle (key-only unforgeability, uf-koa). This means for the named schemes that the

[1] A bijective random oracle is an idealized public bijection that is accessible, in both directions, via oracles; cryptographic constructions that build on such objects include the Even–Mansour blockcipher and the SHA3 hash function.

(restricted) signing oracle is actually redundant. Third, we generalize the notion of *intractable semi-logarithm* from [6] and show that it is equivalent, for some schemes, to key-only unforgeability. In the following we describe these three parts in more detail.

Generic Elgamal Signatures. The GenElgamal signature scheme is defined in the DLP setting relative to a hash function H, a conversion function f, a so-called *defining equation* E, and a set $\mathbb{D}$ that enforces some restrictions on the signature values. See Sect. 3 for the details. Different choices of these parameters lead to different signature schemes, including DSA, ECDSA, GOST, and SM2.

Proving the security of GenElgamal. Consider GenElgamal and assume H is a random oracle. In Sect. 4 we prove that, in this setting, key-only unforgeability implies one-per-message unforgeability. (The latter notion is not only of theoretical interest; as we elaborate in Sect. 2 it is sufficient in many practical scenarios.) This observation can be traced back to Brown [6, II.4.4] for the case of ECDSA, but previously it has not been proved formally. Surprisingly, our security reduction requires a Coron-like partitioning argument [8]. We note that our reduction is not tight but loses a factor of about Q_s (the number of queries to the signing oracle).

Intractable Semi-logarithm. The notion of intractable semi-logarithm was introduced by Brown [6, II.2.2] to analyze the security of ECDSA. The idea is effectively to remove hash function H from the assumption that ECDSA is unforgeable. In brief, a semi-logarithm challenge consists of computing, given g and $X = g^x$, a pair (s, t) such that $t = f((gX^t)^{1/s})$. We formalize and generalize the semi-logarithm assumption in Sect. 5 and show that, in the random oracle model, its hardness is equivalent to the key-only unforgeability of the signature schemes considered in this article (except for SM2).

2 Preliminaries

Notation. For a set $\mathbb{A}$ we write $\mathbb{A}^n$ for the n-fold Cartesian product. We denote random sampling from a finite set $\mathbb{A}$ according to the uniform distribution with $a \xleftarrow{\$} \mathbb{A}$. We use symbol $\xleftarrow{\$}$ also for assignments from randomized algorithms, while we denote assignments from deterministic algorithms and calculations with $\leftarrow$. All algorithms are randomized unless explicitly noted. When using symbols like $\perp$ we mean special symbols that do not appear as elements of sets (e.g., key spaces). Any computation involving $\perp$ results in $\perp$, in particular for every function f we have $f(\perp) = \perp$.

If q is a prime number, we write $\mathbb{Z}_q$ for the field $\mathbb{Z}/q\mathbb{Z}$ and assume the canonic representation of its elements as a natural number in the interval $[0, q-1]$. That is, an element $a \in \mathbb{Z}_q$ is invertible iff $a \neq 0$. We denote prime-order groups with $(\mathbb{G}, g, q)$ where $\mathbb{G}$ is (the description of) a cyclic group, its order $q = |\mathbb{G}|$ is a prime number, and g is a generator such that $\mathbb{G} = \langle g \rangle$. We write 1 for the neutral element of $\mathbb{G}$ and $\mathbb{G}^* = \mathbb{G} \backslash \{1\}$ for the set of its generators.

Our security definitions are game based and expressed via program code. As data structures, besides sets our code may use associative arrays (look-up

tables). We use notation $A[\cdot] \leftarrow \varnothing$ to initialize all cells of an array A to empty. A game G consists of an INIT procedure, one or more procedures to respond to adversary oracle queries, and a FIN procedure. G is executed with an adversary $\mathcal{A}$ as follows: INIT is always run first and its outputs are the inputs to $\mathcal{A}$. Next, the oracle queries of $\mathcal{A}$ are answered by the corresponding procedures of G. Finally, $\mathcal{A}$ calls FIN and terminates. Whenever the Stop command is invoked in a game, the execution of game and adversary is halted and the command's argument is considered the output of the game. We write 'Abort' as a shortcut for 'Stop with 0'. By $G^{\mathcal{A}} \Rightarrow \texttt{out}$ we denote the event that game G executed with $\mathcal{A}$ invokes the Stop command with argument `out`.

SIGNATURE SCHEMES. A *signature scheme* consists of algorithms KGen,Sign, Verify such that: algorithm KGen generates a signing key sk and a verification key pk; on input a signing key sk and a message m algorithm Sign generates a signature σ or the failure indicator $\perp$; on input a verification key pk, a message m, and a candidate signature σ, deterministic algorithm Verify outputs 0 or 1 to indicate rejection and acceptance, respectively. A signature scheme is correct if for all key pairs (sk, pk) created by KGen and all messages m, an invocation of $\mathsf{Sign}(sk, m)$ results in a signature with overwhelming probability, and if it does so then Verify accepts it.

We specify three security notions for signature schemes: uf-cma, uf-cma1, and uf-koa. The standard goal is *unforgeability under chosen-message attack* (uf-cma), meaning that no adversary can produce a valid signature on a fresh message, even if it sees signatures on messages of its choosing. A slightly weaker notion is *one-per-message unforgeability* (uf-cma1) [2,15,25] that adds the restriction that the adversary can see at most one signature per message. The weakest notion considered in this paper is *key-only unforgeability* (uf-koa) where the adversary sees no sample signature but only the verification key. The corresponding security games are in Fig. 1. Note that the uf-cma1 game aborts if the adversary queries the signing oracle a second time on any message, and that in the uf-koa game there is no signing oracle.

Definition 1 (Unforgeability). *For a signature scheme, a forger $\mathcal{F}$ is said to (τ, Q_s, ε)-break* uf-cma (uf-cma1, uf-koa) *security if it runs in at most time τ, poses at most Q_s queries to the* SIGN *oracle, and achieves a forging advantage of $\varepsilon = \Pr[G^{\mathcal{F}} \Rightarrow 1]$, where G is the corresponding game in Fig. 1. (In the* uf-koa *case we require $Q_s = 0$.)*

If the signature scheme is specified in relation to some idealized primitive that is accessed via oracles, we also annotate the maximum number of corresponding queries; for instance, in the random oracle model for a hash function H we use the expression $(\tau, Q_s, Q_H, \varepsilon)$. We always assume that forgers that output a forgery attempt (m^*, σ^*) pose a priori all (public) queries that the verification in FIN *will require.*

Note that, while the uf-cma1 notion is technically weaker than uf-cma security, for many practical applications the former is natural and sufficient. For instance, in Signed-Diffie-Hellman key agreement users exchange messages of

Procedure INIT	**Procedure** FIN(m^*, σ^*)
00 $\mathcal{L} \leftarrow \emptyset$	07 If $m^* \in \mathcal{L}$: Abort
01 $(sk, pk) \xleftarrow{\$} \mathsf{KGen}$	08 If $\mathsf{Verify}(pk, m^*, \sigma^*) = 0$: Abort
02 Return pk	09 Stop with 1
Procedure SIGN(m) (uf-cma)	**Procedure** SIGN(m) (uf-cma1)
⋮	10 If $m \in \mathcal{L}$: Abort
03 $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$	11 $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, m)$
04 If $\sigma = \bot$: Return $\bot$	12 If $\sigma = \bot$: Return $\bot$
05 $\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$	13 $\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$
06 Return σ	14 Return σ

Fig. 1. The vertical space above Line 03 is exclusively for aligning the SIGN oracles of variant uf-cma and of variant uf-cma1 (that adds Line 10). In variant uf-koa the SIGN oracle does not exist.

the form $g^x \parallel \mathsf{Sign}(sk, g^x)$, where exponent x is fresh for each execution and thus no value g^x is ever signed twice. For cases where uf-cma security is not sufficient, [2] propose efficient generic transformations that turn uf-cma1 secure signature schemes into ones secure in the uf-cma sense. Concretely, one possibility is to derandomize the signing algorithm by obtaining the randomness from a secretly keyed function applied to the message.

3 The Generic Elgamal Framework

We recall the abstract signature framework GenElgamal from [23, Sect. 11] that is defined relative to a group $\mathbb{G}$, a hash function H, a conversion function f, and an equation $E(s, h, t, r, x)$ called the *defining equation* of GenElgamal. To the latter is also associated a set $\mathbb{D}$. In GenElgamal, the hash function H is used to hash messages to elements of field $\mathbb{Z}_q$, and the conversion function f is used to transform group elements to elements of $\mathbb{Z}_q$. Intuitively, a signature consists of a solution s of E for values $h = H(m)$, $t = f(g^r)$ where r is the signing randomness, and signing key x. As we will see, to ensure functionality and security, certain such solutions need to be excluded. This is implemented by filtering them out by requiring containedness of corresponding triples (s, h, t) in set $\mathbb{D}$. As it turns out, some standards are overly restrictive on the set of possible signatures (i.e., set $\mathbb{D}$ is specified smaller than it could be; an example is SGenSM2 where $s = 0$ is not allowed). Nevertheless, in this document we stick to the sets specified by the standard documents unless further noted.

Different choices of the defining equation E (and set $\mathbb{D}$) lead to different signature schemes. See Fig. 2 for an overview of classic ones. All these schemes are rooted in the Elgamal signature scheme [10].

Definition 2 (Defining Equation). *Let $\mathbb{D} \subseteq \mathbb{Z}_q^3$ be a set. An equation*

$$E = E(s, h, t, r, x) \text{ over } \mathbb{D} \times (\mathbb{Z}_q^*)^2$$

is said to be defining *(a signature scheme) if E has the form*

$$E(s,h,t,r,x) = C_0(s,h,t) + r\,C_\mathrm{r}(s,h,t) + x\,C_\mathrm{x}(s,h,t),$$

where C_0, C_x are functions $\mathbb{D} \to \mathbb{Z}_q$, and C_r is a function $\mathbb{D} \to \mathbb{Z}_q^*$. With other words, E is defining if it is affine linear in x and r, and E can always be solved for r.

Figure 2 lists possible defining equations together with common names for the corresponding signature schemes. Concretely, we consider all variants of Elgamal signatures mentioned in the Handbook of Applied Cryptography [23], and in addition SM2.[2] Of course there are also other possible choices for E; for example, [17] lists a total of 18 configurations.

Scheme			E	$\mathbb{D}$
GenDSA	(V1)	[20]	$h + tx = rs$	$\mathbb{Z}_q^* \times \mathbb{Z}_q \times \mathbb{Z}_q^*$
GenGOST	(V3)	[9]	$hr + tx = s$	$\mathbb{Z}_q^* \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*$
SGenSM2		[19]	$h + r + t = sx$	$\mathbb{D}_{\mathrm{SM}}(t)$
GenAMV	(V2)	[1]	$h = rt + sx$	$\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q^*$
GenHarn	(V6)	[16]	$hsx + r = st$	$\mathbb{Z}_q^* \times \mathbb{Z}_q \times \mathbb{Z}_q^*$
no name	(V4)	[23]	$hx + rt = s$	$\mathbb{Z}_q \times \mathbb{Z}_q \times \mathbb{Z}_q^*$
no name	(V5)	[23]	$hr + t = sx$	$\mathbb{Z}_q \times \mathbb{Z}_q^* \times \mathbb{Z}_q^*$

Fig. 2. Defining equations of a selection of established signature schemes. The variant number (Vi) refers to [23, Table 11.5]. $\mathbb{D}_{\mathrm{SM}}(t)$ is defined as $\{(s,h,t) \in \mathbb{Z}_q^* \times \mathbb{Z}_q^2 : t+h \neq 0, s-t-h \neq 0\}$.

Definition 3 (Signing and Verification Function). *Let E be a defining equation. Then we define the* signing function $\mathsf{S}^E(h,t,r,x) = \mathsf{S}^E_x(h,t,r)$ *as follows: if there exists a unique s such that $E(s,h,t,r,x)$ is satisfied, S^E returns s; otherwise, the function returns $\perp$.*

Further, we define the verification function $\mathsf{V}^E(g,s,h,t,x) = \mathsf{V}^E_{g,x}(s,h,t)$ *with respect to a prime-order group $(\mathbb{G},g,q)$ as follows: if r is the (unique) solution of $E(s,h,t,r,x)$ then V^E returns g^r.*

Note that the affine linear form of E makes it possible to efficiently evaluate V^E given just s,h,t,g^x, i.e., without knowing x explicitly.

Definition 4 (GenElgamal Framework). *Let $(\mathbb{G},g,q)$ be a prime-order group, $\mathbb{D} \subseteq \mathbb{Z}_q^3$ a set, and $H\colon \{0,1\}^* \to \mathbb{Z}_q$ a hash function. Let further $f\colon \mathbb{G}^* \to \mathbb{Z}_q$ be a function and E a defining equation as in Definition 2. Then* GenElgamal *(relative to $E,\mathbb{G},H,f,\mathbb{D}$) is defined by the algorithms of Fig. 3.*

[2] More precisely, we consider SGenSM2 which is an equivalent variant of SM2. Concretely, $(\hat{s},\hat{t})$ is a valid SM2 signature on a message m for the verification key $\hat{X}$ if and only if $(s,t) = (\hat{s}+\hat{t}, \hat{t} - H(m))$ is a valid SGenSM2 signature on m for the verification key $X = g\hat{X}$. As all these transformations are public and reversible, the functionality and security of SM2 and SGenSM2 are the same.

Proc KGen	**Proc** Sign(sk, m)	**Proc** Verify$(pk, m, (s,t))$
00 $x \xleftarrow{\$} \mathbb{Z}_q^*$; $X \leftarrow g^x$	03 $r \xleftarrow{\$} \mathbb{Z}_q$; $R \leftarrow g^r$	11 $h \leftarrow H(m)$
01 $sk := x$; $pk := X$	04 If $R = 1$: Return $\bot$	12 If $(s,h,t) \notin \mathbb{D}$:
02 Return (sk, pk)	05 $t \leftarrow f(R)$	13 Return 0
	06 $h \leftarrow H(m)$	14 $\hat{R} \leftarrow \mathsf{V}^E_{g,x}(s,h,t)$
	07 $s \leftarrow \mathsf{S}^E_x(h,t,r)$	15 If $\hat{R} = 1$: Return 0
	08 If $(s,h,t) \notin \mathbb{D}$:	16 $\hat{t} \leftarrow f(\hat{R})$
	09 Return $\bot$	17 If $t \neq \hat{t}$: Return 0
	10 Return (s,t)	18 Return 1

Fig. 3. The GenElgamal signature scheme with defining equation E. Functions S^E and V^E are as in Definition 3. If S^E returns $\bot$ in Line 07 then Sign returns $\bot$ in Line 09.

We define a notion of *simulatability* that will be used in the GenElgamal security proof (in Sect. 4). It captures the fact that, in the random oracle model, it is possible to simulate (almost) correctly distributed GenElgamal signatures without knowledge of the signing key.

Definition 5 (δ**-Simulatability**). *Let* $(E, \mathbb{G}, H, f, \mathbb{D})$ *be an instantiation of* GenElgamal *as in Definition 4. We say the scheme is* δ-simulatable *if there exists a function* $\mathsf{Sim}^E \colon \mathbb{Z}_q^3 \to \mathbb{Z}_q^2 \cup \{\bot\}$ *that is computable in about the same time as* S^E *such that for all* $x \in \mathbb{Z}_q^*$ *the statistical distance between the outputs of the two protocols depicted in Fig. 4 is at most* δ.

Protocol $\mathsf{P}_{\mathrm{real}}(x)$	**Protocol** $\mathsf{P}_{\mathrm{sim}}(g^x)$
00 $r \xleftarrow{\$} \mathbb{Z}_q$	08 $a, b \xleftarrow{\$} \mathbb{Z}_q$
01 $R \leftarrow g^r$	09 $R \leftarrow X^a g^b$
02 If $R = 1$: Return $\bot$	10 If $R = 1$: Return $\bot$
03 $t \leftarrow f(R)$	11 $t \leftarrow f(R)$
04 $h \xleftarrow{\$} \mathbb{Z}_q$	12 $(s,h) \leftarrow \mathsf{Sim}^E(a,b,t)$
05 $s \leftarrow \mathsf{S}^E_x(h,t,r)$	⋮
06 If $(s,h,t) \notin \mathbb{D}$: Return $\bot$	13 If $(s,h,t) \notin \mathbb{D}$: Return $\bot$
07 Return (s,h,t)	14 Return (s,h,t)

Fig. 4. Simulatability of an instantiation of GenElgamal. If Sim^E outputs $\bot$ in Line 12 then $\mathsf{P}_{\mathrm{sim}}$ outputs $\bot$ in Line 13. The vertical space between Lines 12 and 13 is exclusively for aligning the two protocols.

Lemma 1. *All of the instantiations of* GenElgamal *described in Fig. 2 are* δ*-simulatable with* $\delta \leq 2/q$.

Proof. Consider any of the instantiations. Let $x \in \mathbb{Z}_q^*$ be arbitrary. In $\mathsf{P}_{\mathrm{sim}}$ the random value r is implicitly computed in the exponent as $ax + b$ and by choice

of a and b uniformly distributed on $\mathbb{Z}_q$, so the t-values in both protocols are distributed identically.

Next, we want to show that for fixed r, t, x the value a is almost always a function in h and vice versa. To this end we show that for each instantiation there exist sets $\mathbb{A}, \mathbb{H} \subseteq \mathbb{Z}_q$ (depending on $r, t = f(g^r), x$) with $|\mathbb{H}| \geq q - 2$ and a bijection $\pi_{x,r} \colon \mathbb{H} \to \mathbb{A}$. The bijection and its inverse function can be computed directly from the respective defining equation, see Fig. 5. Note that $\pi_{x,r}^{-1}$ actually is a function of a, b, t, but for fixed x, r the value of b is uniquely determined by the choice of a as $b = r - ax$ and the value of t is uniquely determined as $t = f(g^r)$. Now when sampling $a \xleftarrow{\$} \mathbb{Z}_q$ and computing h as $\pi_{x,r}^{-1}(a, r - ax, f(g^r))$ in $\mathsf{P}_{\mathrm{sim}}(x)$ (setting $\pi_{x,r}^{-1}(a, r - ax, f(g^r)) = \perp$ for $a \notin \mathbb{A}$, which happens with probability at most $2/q$ since $|\mathbb{Z}_q \backslash \mathbb{A}| \leq 2$) instead of directly sampling h uniformly random from $\mathbb{Z}_q$ in $\mathsf{P}_{\mathrm{real}}(x)$, the statistical distance between the h-values is at most $2/q$.

Scheme	$\mathbb{H}$	$\mathbb{A}$	$\pi_{x,r}$	$\pi_{x,r}^{-1}$	$\xi_{x,r}$	δ
GenDSA	$\mathbb{Z}_q \setminus \{-xt\}$	$\mathbb{Z}_q^*$	$rt/(h + xt)$	bt/a	t/a	$1/q$
GenGOST	$\mathbb{Z}_q^*$	$\mathbb{Z}_q^*$	$-1/h$	$-1/a$	$-b/a$	$1/q$
SGenSM2	$\mathbb{Z}_q \setminus \{-t - r\}$	$\mathbb{Z}_q^*$	$(h + r + t)/x$	$-(b + t)$	a	$1/q$
GenAMV	$\mathbb{Z}_q \setminus \{rt\}$	$\mathbb{Z}_q^*$	$(h - rt)/tx$	bt	$-at$	$1/q$
GenHarn	$\mathbb{Z}_q^* \setminus \{t/x\}$	$\mathbb{Z}_q^* \setminus \{r/x\}$	$hr/(hx - t)$	$-at/b$	b/t	$2/q$
(V4)	$\mathbb{Z}_q^*$	$\mathbb{Z}_q^*$	$-h/t$	$-at$	bt	$1/q$
(V5)	$\mathbb{Z}_q^* \setminus \{-t/r\}$	$\mathbb{Z}_q^* \setminus \{r/x\}$	$(hr + t)/hx$	$-t/b$	$-at/b$	$2/q$

Fig. 5. Sets $\mathbb{H}$ and $\mathbb{A}$ and functions $\pi_{x,r}(h)$, $\pi_{x,r}^{-1}(a, b, t)$, and $\xi_{x,r}(a, b, t)$ for the schemes from Fig. 2. We write $t = f(g^r)$. The last column shows the δ-values for the simulatability of the instantiation (see Definition 5).

Now once x, a, b, t, h are fixed, since the defining equation has to hold, s can be computed deterministically by a function $\xi_{x,r}$, also displayed in Fig. 5. Note that both $\pi_{x,r}^{-1}$ and $\xi_{x,r}$ can be computed without explicit knowledge of x, r for all of the instantiations. So if we set

$$\mathsf{Sim}^E(a, b, t) = (\xi_{x,r}(a, b, t), \pi_{x,r}^{-1}(a, b, t)),$$

the statistical distance between the outputs of the two protocols from Fig. 4 is at most $2/q$. □

4 Security of GenElgamal in the ROM

We examine the security of GenElgamal, showing that if the hash function H is modeled as a random oracle, key-only unforgeability implies one-per-message unforgeability. This was already suggested in [6, II.4.4] for the case of GenDSA, but no formal treatment was given. We here provide a formal statement and a proof for the general case. Interestingly, our argument involves Coron-type partitioning [8].

Theorem 1. *Let $E, \mathbb{G}, H, f, \mathbb{D}$ be a δ-simulatable instantiation of* GenElgamal. *Then if H is modeled as a random oracle, for every forger $\mathcal{F}$ that $(\tau, Q_s, Q_H, \varepsilon)$-breaks the one-per-message unforgeability of this instantiation there also exists a forger $\mathcal{F}'$ that $(\tau', 0, Q_H, \varepsilon')$-breaks the key-only unforgeability of this instantiation, where*

$$\varepsilon' \geq \varepsilon/(e^2(Q_s+1)) - Q_s\delta \qquad \textit{and} \qquad \tau' = \tau + \mathcal{O}(Q_H).$$

Proof. Let $\mathcal{F}$ be a forger that $(\tau, Q_s, Q_H, \varepsilon)$-breaks the one-per-message unforgeability of the scheme under consideration. Let **Game $\mathbf{G}_0$** be the standard uf-cma1 game with the algorithms of Fig. 3 plugged in and an additional random oracle RO for H that is implemented by lazy sampling (see Fig. 6). We assume without loss of generality that $\mathcal{F}$ queries RO on m before calling Sign or Fin involving the same message. We have

$$\Pr[G_0^{\mathcal{F}} \Rightarrow 1] = \varepsilon.$$

The idea of the reduction is that we respond to each hash query $\mathrm{RO}(m)$ by selecting the hash value in a specific though uniform way (such that we can simulate signatures on m), except for the value of m^*, which we want to forward to the random oracle RO^* of the uf-koa security game in a reduction later. But m^* is not yet known at the time of simulating the hash queries, so in **Game $\mathbf{G}_1$** (see Fig. 6) we apply the partitioning technique from [8] and toss a biased coin that takes value 0 with probability $Q_s/(Q_s+1)$ and value 1 with probability

Procedure Init	**Procedure** $\mathrm{RO}(m)$
00 $\mathcal{L} \leftarrow \emptyset$	14 If $H[m] \neq \varnothing$:
01 $H[\cdot] \leftarrow \varnothing$; $c[\cdot] \leftarrow \varnothing$	15 Return $H[m]$
02 $x \xleftarrow{\$} \mathbb{Z}_q^*$; $X \leftarrow g^x$	16 $c[m] \xleftarrow{\$} \mathrm{Ber}(\gamma)$ (G_1)
03 Return X	17 $h \xleftarrow{\$} \mathbb{Z}_q$
	18 $H[m] \leftarrow h$
Procedure Sign(m)	19 Return h
04 If $m \in \mathcal{L}$: Abort	
05 If $c[m] \neq 0$: Abort (G_1)	**Procedure** Fin$(m^*, (s^*, t^*))$
06 $r \xleftarrow{\$} \mathbb{Z}_q$; $R \leftarrow g^r$	20 If $m^* \in \mathcal{L}$: Abort
07 If $R = 1$: Return $\perp$	21 If $c[m^*] \neq 1$: Abort (G_1)
08 $t \leftarrow f(R)$	22 $h^* \leftarrow H[m^*]$
09 $h \leftarrow H[m]$	23 If $(s^*, h^*, t^*) \notin \mathbb{D}$: Abort
10 $s \leftarrow \mathsf{S}_x^E(h, t, r)$	24 $R^* \leftarrow \mathsf{V}_{g,x}^E(s^*, h^*, t^*)$
11 If $(s, h, t) \notin \mathbb{D}$: Return $\perp$	25 If $R^* = 1$: Abort
12 $\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$	26 If $f(R^*) \neq t^*$: Abort
13 Return (s, t)	27 Stop with 1

Fig. 6. Games G_0 and G_1. Ber is the Bernoulli distribution with bias $\gamma = 1/(Q_s+1)$, i.e., in Line 16 $c[m]$ takes the value 1 with probability $1/(Q_s+1)$. Note that Line 20 is redundant in G_1.

$\gamma = 1/(Q_s + 1)$ for every queried message, and we hope that it takes the value 0 for all messages used in signature queries and the value 1 for m^*.

We now analyze the probability that one of the coins takes an unwanted value, i.e., the probability of an abort in Lines 05 and 21. To do this, we consider the complementary probability. Since for all messages m, $c[m]$ is distributed according to the Bernoulli distribution $\mathrm{Ber}(\gamma)$ with $\gamma = 1/(Q_s + 1)$ and independently of all other coins, the probability that no abort happens in these lines is

$$(1-\gamma)^{Q_s}\gamma \geq (1 - 1/Q_s)^{Q_s}(1/(Q_s+1)) \geq 1/e^2(Q_s+1),$$

where the last inequality is a standard result in calculus and holds for $Q_s \geq 2$. The case $Q_s = 1$ is trivial. It follows that

$$\Pr[G_0^{\mathcal{F}} \Rightarrow 1] \leq e^2(Q_s+1)\Pr[G_1^{\mathcal{F}} \Rightarrow 1].$$

In **Game $\mathbf{G}_2$** (see Fig. 7) we introduce two changes: (a) when processing a random oracle query on a message m, a signature for m is precomputed and stored, and (b) the way of signing messages is changed so that signatures are generated without knowing the signing key. Note that change (a) is possible only because the SIGN oracle may be queried on each message at most once. Change (b) exploits the assumed simulatability (see Definition 5) of GenElgamal.

Procedure INIT
00 $\mathcal{L} \leftarrow \emptyset$
01 $H[\cdot] \leftarrow \varnothing$; $c[\cdot] \leftarrow \varnothing$
02 $\sigma[\cdot] \leftarrow \varnothing$
03 $x \xleftarrow{\$} \mathbb{Z}_q^*$; $X \leftarrow g^x$
04 Return X

Procedure SIGN(m)
05 If $m \in \mathcal{L}$: Abort
06 If $c[m] \neq 0$: Abort
07 $\mathcal{L} \leftarrow \mathcal{L} \cup \{m\}$
08 Return $\sigma[m]$

Procedure FIN($m^*, (s^*, t^*)$)
as in G_1 (Fig. 6)

Procedure RO(m)
09 If $H[m] \neq \varnothing$: Return $H[m]$
10 $c[m] \xleftarrow{\$} \mathrm{Ber}(\gamma)$
11 $h \xleftarrow{\$} \mathbb{Z}_q$
12 If $c[m] = 0$:
13 $\quad a, b \xleftarrow{\$} \mathbb{Z}_q$
14 $\quad R \leftarrow X^a g^b$
15 $\quad$ If $R = 1$:
16 $\quad\quad \sigma[m] \leftarrow \perp$; Goto Line 22
17 $\quad t \leftarrow f(R)$
18 $\quad (s, h) \leftarrow \mathsf{Sim}^E(a, b, t)$
19 $\quad$ If $(s, h, t) \notin \mathbb{D}$:
20 $\quad\quad \sigma[m] \leftarrow \perp$; Goto Line 22
21 $\quad \sigma[m] \leftarrow (s, t)$
22 $H[m] \leftarrow h$
23 Return h

Fig. 7. Game G_2

We argue that the adversary can distinguish G_1 and G_2 with probability at most $Q_s\delta$. To see this, note that change (a) is a pure rewriting step and does not influence the output of the game. Concerning change (b), consider first the case that the adversary queries SIGN or RO on a message m with $c[m] = 1$. For the random oracle, the response h is picked uniformly at random in Line 11, and the signing oracle aborts, so the distribution is exactly as in G_1.

Consider next the case that the adversary queries one of the oracles on a message m with $c[m] = 0$. Observe then that Lines 06 to 11 in G_1 correspond exactly to the protocol P_{real} from Fig. 4, and Lines 13 to 20 in G_2 correspond exactly to the protocol P_{sim}. Thus, switching the way of computing signatures introduces, for each call to the signing oracle, a statistical distance between the two games that is bounded by δ. We obtain

$$\left|\Pr[G_1^{\mathcal{F}} \Rightarrow 1] - \Pr[G_2^{\mathcal{F}} \Rightarrow 1]\right| \leq Q_s\delta.$$

Now construct a uf-koa forger $\mathcal{F}'$ against GenElgamal in the random oracle model as in Fig. 8.

Procedure INIT replace Line 03 with 03 $X \xleftarrow{\$}$ INIT*	**Procedure** SIGN(m) as in G_2 (Fig. 7)
Procedure RO(m) replace Line 11 with 11 $h \leftarrow \text{RO}^*(m)$	**Procedure** FIN$(m^*, (s^*, t^*))$ replace Lines 22–27 in Fig. 6 with 22 Invoke FIN$^*(m^*, (s^*, t^*))$

Fig. 8. Construction of uf-koa forger $\mathcal{F}'$ from $\mathcal{F}$ by changing Game G_2 as specified. INIT*, RO*, and FIN* are the procedures from the uf-koa security game run by $\mathcal{F}'$. Procedure SIGN is as in Game G_2.

The coin tosses in Line 10 of Fig. 7 ensure that $\mathcal{F}'$ only has to provide signatures on messages for which it programmed the random oracle itself; it thus simulates the signing procedure of G_2 perfectly. Further, the coin tosses guarantee that the forgery is consistent with RO*, so $\mathcal{F}'$ wins its game exactly if $\mathcal{F}$ produces a valid forgery. This means that

$$\Pr[G_2^{\mathcal{F}} \Rightarrow 1] = \varepsilon',$$

and the statement follows. □

5 The Semi-Logarithm Problem

We formalize and generalize the notion of intractable *semi-logarithm problem* (SLP), a notion introduced by Brown for the analysis of signature schemes.

His motivation for studying the SLP is "to isolate the role of the hash function and the group in analyzing the security of ECDSA" [6, p. 25]. Effectively, the SLP is a number-theoretic hardness assumption related to the search problem of finding a valid GenElgamal signature for a (unknown) message m with hash value $H(m) = 1$.

As we show, the key-only unforgeability of an instantiation of GenElgamal is characterized by the intractability of the corresponding variant of the semi-logarithm problem (in the random oracle model), potentially establishing a simplified target for cryptanalysis. Note that a suitable SLP variant does not exist for all GenElgamal instantiations: for SM2 there is apparently no corresponding SLP definition.

Definition 6. *Let $(\mathbb{G}, g, q)$ be a prime-order group and let $f\colon \mathbb{G}^* \to \mathbb{Z}_q$ and $\rho_0, \rho_1\colon \mathbb{Z}_q^2 \to \mathbb{Z}_q$ be functions. We say that an algorithm $\mathcal{I}$ (τ, ε)-breaks the* semi-logarithm problem (SLP) *in $\mathbb{G}$ with respect to f, ρ_0, ρ_1 if it runs in time at most τ and achieves probability*

$$\varepsilon = \Pr[X \xleftarrow{\$} \mathbb{G}; (u, v) \xleftarrow{\$} \mathcal{I}(g, X) : v = f(g^{\rho_0(u,v)} X^{\rho_1(u,v)})].$$

Definition 7. *Let $E = E(s, h, t, r, x)$ be a defining equation with corresponding set $\mathbb{D}$ (see Definition 2). We say that E is h-decomposable (with respect to $\mathbb{D}$) if there exist functions*

$$\eta_0, \eta_1\colon \mathbb{Z}_q \to \mathbb{Z}_q \qquad \text{and} \qquad \rho_0, \rho_1\colon \mathbb{Z}_q^2 \to \mathbb{Z}_q$$

such that $\eta_0(h), \eta_1(h) \neq 0$ if $h \neq 0$ and

$$r = \eta_0(h)\rho_0(s, t) + x\,\eta_1(h)\rho_1(s, t)$$

for all $(s, h, t) \in \mathbb{D}$ and $r, x \in \mathbb{Z}_q^*$ satisfying $E(s, h, t, r, x)$.

All defining equations from Fig. 2, except for SGenSM2, are h-decomposable. The corresponding components $\eta_0, \rho_0, \eta_1, \rho_1$ are listed in Fig. 9.

Scheme	$\eta_0(h)$	$\rho_0(s,t)$	$\eta_1(h)$	$\rho_1(s,t)$
GenDSA (V1)	h	$1/s$	1	t/s
GenGOST (V3)	$1/h$	s	$-1/h$	t
GenAMV (V2)	h	$1/t$	-1	s/t
GenHarn (V6)	1	st	$-h$	s
no name (V4)	1	s/t	$-h$	$1/t$
no name (V5)	$-1/h$	t	$1/h$	s

Fig. 9. Components $\eta_0, \rho_0, \eta_1, \rho_1$ of the h-decomposable defining equations from Fig. 2.

Theorem 2. *Let $(\mathbb{G}, g, q)$ be a prime-order group, let E be a defining equation with corresponding set $\mathbb{D}$, and let $f\colon \mathbb{G}^* \to \mathbb{Z}_q$ and $H\colon \{0,1\}^* \to \mathbb{Z}_q$ be functions. If E is h-decomposable with functions ρ_0, ρ_1, and H is modeled as a random oracle, then the semi-logarithm problem in $\mathbb{G}$ with respect to f, ρ_0, ρ_1 is non-tightly equivalent to the key-only unforgeability of* GenElgamal *when instantiated with $E, \mathbb{G}, H, f, \mathbb{D}$.*

More precisely, for any adversary $\mathcal{I}$ that (τ, ε)-breaks SLP, there exists a forger $\mathcal{F}$ that (τ', ε)-breaks the key-only unforgeability of GenElgamal*, where $\tau \approx \tau'$.*

Conversely, for any forger $\mathcal{F}$ that (τ, Q_H, ε)-breaks the key-only unforgeability of GenElgamal*, there exists an adversary $\mathcal{I}$ that $(\tau', \varepsilon/Q_H - 1/q)$-breaks SLP, where $\tau' \approx \tau$ and Q_H is the number of random oracle queries posed by $\mathcal{F}$.*

Proof. Given an adversary $\mathcal{I}$ that (τ, ε)-breaks SLP, we construct a forger $\mathcal{F}$ that (τ', ε)-breaks key-only unforgeability of GenElgamal, for any hash function H. (For the particular case of ECDSA, this result is due to Brown [6].) On input, $\mathcal{F}$ obtains g, X (from pk). It picks any message m (independently of X) such that $H(m) \neq 0$, computes $h \leftarrow H(m)$, $g' \leftarrow g^{\eta_0(h)}$, and $X' \leftarrow X^{\eta_1(h)}$, and lets $\mathcal{I}$ compute a semi-logarithm as per $(u, v) \xleftarrow{\$} \mathcal{I}(g', X')$. Then (u, v) is a valid signature on m (with respect to g, X). Indeed, since E is h-decomposable, by definition of $\mathsf{V}^E_{g,x}$ (see Definition 3) it holds that in Verify (see Line 14 in Fig. 3) we have

$$\hat{R} = \mathsf{V}^E_{g,x}(u, h, v) = g^{\eta_0(h)\rho_0(u,v)} X^{\eta_1(h)\rho_1(u,v)} = (g')^{\rho_0(u,v)} (X')^{\rho_1(u,v)},$$

and thus $f(\hat{R}) = v$.

Let now $\mathcal{F}$ be a forger that (τ, Q_H, ε)-breaks the key-only unforgeability of GenElgamal. We construct an adversary $\mathcal{I}$ against SLP from it. On input of (g, X), it draws $a \xleftarrow{\$} \mathbb{Z}_q$, aborts if $a = 0$, sets $g' \leftarrow g^{1/\eta_0(a)}$ and $X' \leftarrow X^{1/\eta_1(a)}$, and starts $\mathcal{F}$ on input $pk = (g', X')$. If m^* denotes the message on which $\mathcal{F}$ forges a signature, we assume w.l.o.g. that $\mathcal{F}$ queries $H(m^*)$ before outputting the latter. $\mathcal{I}$ initially guesses the index $j \in \{1, \ldots, Q_H\}$ of the corresponding query to H. It then responds to the jth random oracle query by programming it via $H(m_j) \leftarrow a$, and answers all other queries with uniform values. Once $\mathcal{F}$ outputs its forgery $(m^*, (s, t))$, adversary $\mathcal{I}$ forwards (s, t) to its own challenger. Since E is h-decomposable and $g = (g')^{\eta_0(H(m^*))}$ and $X = (X')^{\eta_1(H(m^*))}$, it holds that

$$\begin{aligned} g^{\rho_0(s,t)} X^{\rho_1(s,t)} &= ((g')^{\eta_0(H(m^*))})^{\rho_0(s,t)} ((X')^{\eta_1(H(m^*))})^{\rho_1(s,t)} \\ &= \mathsf{V}^E_{g',x'}(s, H(m^*), t), \end{aligned}$$

where $x' = \log_{g'} X'$. That is, $\mathcal{I}$ wins in the SLP game if it didn't abort when sampling a, its guess for index j was correct, and $\mathcal{F}$ forges successfully. □

Acknowledgments. The first author was supported by DFG SPP 1736 Big Data. The second author was supported in part by ERC Project ERCC (FP7/615074) and by DFG SPP 1736 Big Data. The third author was supported in part by ERC Project ERCC (FP7/615074).

References

1. Agnew, G., Mullin, R., Vanstone, S.: Improved digital signature scheme based on discrete exponentiation. Electron. Lett. **26**(14), 1024–1025 (1990)
2. Bellare, M., Poettering, B., Stebila, D.: From identification to signatures, tightly: a framework and generic transforms. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 435–464. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53890-6_15

3. Brickell, E., Pointcheval, D., Vaudenay, S., Yung, M.: Design validations for discrete logarithm based signature schemes. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 276–292. Springer, Heidelberg (2000). doi:10.1007/978-3-540-46588-1_19
4. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Cryptology ePrint Archive, Report 2002/026 (2002). http://eprint.iacr.org/2002/026
5. Brown, D.R.L.: Generic groups, collision resistance, and ECDSA. Des. Codes Crypt. **35**(1), 119–152 (2005)
6. Brown, D.R.L.: On the provable security of ECDSA. In: Blake, I.F., Seroussi, G., Smart, N.P. (eds.) Advances in Elliptic Curve Cryptography, pp. 21–40. Cambridge University Press, Cambridge (2005). doi:10.1017/CBO9780511546570.004
7. Brown, D.R.L.: One-up problem for (EC)DSA. Cryptology ePrint Archive, Report 2008/286 (2008). http://eprint.iacr.org/2008/286
8. Coron, J.-S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_14
9. Dolmatov, V., Degtyarev, A.: GOST R 34.10-2012: Digital Signature Algorithm. RFC 7091 (Informational), December 2013. http://www.ietf.org/rfc/rfc7091.txt
10. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_2
11. Fersch, M., Kiltz, E., Poettering, B.: On the provable security of (EC)DSA signatures. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 16, Vienna, Austria, 24–28 October 2016, pp. 1651–1662. ACM Press (2016)
12. Fersch, M., Kiltz, E., Poettering, B.: On the one-per-message unforgeability of (EC)DSA and its variants. Cryptology ePrint Archive, Report 2017/890 (2017). http://eprint.iacr.org/2017/890
13. García, C.P., Brumley, B.B., Yarom, Y.: Make sure DSA signing exponentiations really are constant-time. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, Vienna, Austria, 24–28 October 2016, pp. 1639–1650. ACM Press (2016)
14. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E., Yarom, Y.: ECDSA key extraction from mobile devices via nonintrusive physical side channels. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016, Vienna, Austria, 24–28 October 2016, pp. 1626–1638. ACM Press (2016)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC, Victoria, British Columbia, Canada, 17–20 May 2008, pp. 197–206. ACM Press (2008)
16. Harn, L.: New digital signature scheme based on discrete logarithm. Electron. Lett. **30**(5), 396–398 (1994)
17. Harn, L., Xu, Y.: Design of generalised ElGamal type digital signature schemes based on discrete logarithm. Electron. Lett. **30**(24), 2025–2026 (1994)
18. Howgrave-Graham, N., Smart, N.P.: Lattice attacks on digital signature schemes. Des. Codes Crypt. **23**(3), 283–290 (2001)
19. ISO/IEC 11889:2015: Information technology—Trusted Platform Module library (2013). https://www.iso.org/
20. Kerry, C.F., Gallagher, P.D.: FIPS PUB 186–4 Federal Information Processing Standards publication: Digital Signature Standard (DSS) (2013). doi:10.6028/NIST.FIPS.186-4

21. Leadbitter, P.J., Page, D., Smart, N.P.: Attacking DSA under a repeated bits assumption. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 428–440. Springer, Heidelberg (2004). doi:10.1007/978-3-540-28632-5_31
22. Malone-Lee, J., Smart, N.P.: Modifications of ECDSA. In: Nyberg, K., Heys, H. (eds.) SAC 2002. LNCS, vol. 2595, pp. 1–12. Springer, Heidelberg (2003). doi:10.1007/3-540-36492-7_1
23. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. The CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, Boca Raton (1997). 2000 N.W. Corporate Blvd., FL 33431–9868, USA
24. Nguyen, P.Q., Shparlinski, I.: The insecurity of the elliptic curve digital signature algorithm with partially known nonces. Des. Codes Crypt. **30**(2), 201–217 (2003)
25. Poettering, B., Stebila, D.: Double-authentication-preventing signatures. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 436–453. Springer, Cham (2014). doi:10.1007/978-3-319-11203-9_25
26. Pointcheval, D., Vaudenay, S.: On provable security for digital signature algorithms. Technical report LIENS-96-17, LIENS (1996)
27. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997). doi:10.1007/3-540-69053-0_18
28. Stern, J., Pointcheval, D., Malone-Lee, J., Smart, N.P.: Flaws in applying proof methodologies to signature schemes. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 93–110. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_7
29. Vaudenay, S.: Hidden collisions on DSS. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 83–88. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_7
30. Vaudenay, S.: The security of DSA and ECDSA. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 309–323. Springer, Heidelberg (2003). doi:10.1007/3-540-36288-6_23
31. Zhang, Z., Yang, K., Zhang, J., Chen, C.: Security of the SM2 signature scheme against generalized key substitution attacks. In: Chen, L., Matsuo, S. (eds.) SSR 2015. LNCS, vol. 9497, pp. 140–153. Springer, Cham (2015). doi:10.1007/978-3-319-27152-1_7