Evaluation Report on the **ECDSA** signature scheme

Jacques Stern

1 Introduction

This document is an evaluation of the ECDSA signature scheme. Our work is based on the analysis of various documents [1, 32, 10, 11], which provide the specification of the scheme, as well as on various research papers related to the scheme. Among these research papers are the original work of Poincheval and Stern [33, 34] investigating the security of El Gamal-like signatures, and the series of papers related to the so-called generic model [29, 45, 27, 6]. Of particular interest to the present report is [6], which applies the generic model to argue towards the security of ECDSA.

The present report is organized as follows: firstly, we briefly review the cryptographic primitive on which the signature scheme relies, namely the discrete logarithm problem on elliptic curves (ECDL), and analyze the various algorithms that are currently known to solve the problem. We specifically address the case of curves of special form, since some of the recommended examples proposed in the standard [11] are Koblitz curves. Next, we give formal definitions for signature schemes and recall the strongest security notion that is now mandatory for signature schemes: security against existential forgery under adaptive chosen-message attacks. We also provide a short history of the various signature schemes based on the discrete logarithm problem, and of the various strategies used to support their security: the random oracle model and the generic model. This allows us to prove the security of the *generic* version of ECDSA, against adaptive chosen-message attacks. We finally analyze the meaning of this proof, with respect to the actual ECDSA signature scheme. In particular, we discuss whether one can derive practical implications, notably in terms of key sizes. This is as requested by IPA.

2 ECDSA and the Elliptic Curve Discrete Logarithm Problem

2.1 General setting

The discrete logarithm problem in a finite group $\mathcal{G}$ can be stated as follows: compute x from $\mathbf{g}$ and $\mathbf{y} = x \cdot \mathbf{g}$. Integer x is called the discrete logarithm of $\mathbf{y}$ in base $\mathbf{g}$, $x = \log_{\mathbf{g}}(\mathbf{y})$. We will restrict our attention to the case where $\mathcal{G}$ is a finite cyclic group $\mathcal{G}$ of prime order q and $\mathbf{g}$ is a generator of $\mathcal{G}$. In the above, we use bold letters to denote elements of $\mathcal{G}$, so that no confusion arises with scalars, such as x. We write the group operation in additive notation: $x \cdot \mathbf{g}$ is simply $x \cdot \mathbf{g} = \mathbf{g} + \ldots + \mathbf{g}$, where $\mathbf{g}$ is repeated x times. Examples of cryptographic interest are the subgroup of order q of $(\mathbb{Z}_p^\star, \times)$ consisting of elements x in $\mathbb{Z}_p^\star$ such that $x^{\frac{p-1}{q}} = 1 \bmod p$, where q is a large prime factor of $p-1$, and subgroups of prime order of an elliptic curve.

ECDSA is based on the hardness of the discrete logarithm problem over an elliptic curve. The signature scheme uses elliptic curves E over some prime field $\mathbb{F}_p$, $|p| = m$ or elliptic curves over $\mathbb{F}_{2^m}$. In the first case, possible values for m are

$$\{112, 128, 160, 192, 224, 256, 384, 521\}$$

and, in the second case:

$$\{113, 131, 163, 193, 233, 283, 409, 571\}$$

Once the curve E has been chosen, a base point G is chosen on E, which generates a subgroup of order q, such that, denoting by $\#(E)$ the number of points in the curve and defining the *cofactor* h by $h = \#(E)/q$, inequality $h \leq 4$ holds. Examples are given in [11].

A detailed description of ECDSA will be provided later, as well as a security argument that relates the scheme to the ECDL. Therefore, in order to estimate whether the specific restrictions on the curve and the suggested parameters offer a wide security margin, it is useful to review the performances of the various algorithms known for the ECDLP. We will distinguish between exponential algorithms, whose running time depends on the size of the group and subexponential algorithms, which apply to specific classes of weak curves.

2.2 Exponential algorithms for the ECDL

2.2.1 Pollard's ρ-method

The best algorithm known to date for solving the DLP in any given group $\mathcal{G}$ is the Pollard ρ-method from [36] which takes computing time equivalent to about $\sqrt{\pi n/2}$

group operations. In 1993, van Oorschot and Wiener in [49], showed how the Pollard ρ-method can be parallelized so that, if t processors are used, then the expected number of steps by each processor before a discrete logarithm is obtained is $\simeq \frac{\sqrt{\pi n/2}}{t}$. In order to compute the discrete logarithm of Y in base G, each processor computes a kind of random walk within elements of the form $a \cdot G + b \cdot Y$, selecting X_{i+1} through one of the three following rules

1. set $X_{i+1} = G + X_i$
2. set $X_{i+1} = 2 \cdot X_i$
3. set $X_{i=1} = Y + X_i$

Decisions on which rule to apply are made through a random-looking but deterministic computation, using e.g. hash values. So-called *distinguished* points X_i are stored together with their representation $X_i = a_i \cdot G + b_i \cdot Y$ in a list that is common to all processors. When a collision occurs in the list, the requested discrete logarithm becomes known.

The progress of such algorithms is well documented. In April 2000, the solution to the ECC2K-108 challenge from Certicom [9] led to the computation of a discrete logarithm in a group with 2^{109} elements (see [21]). This is one of the largest effort ever devoted to a public-key cryptography challenge. The amount of work required to solve the ECC2K-108 challenge was about 50 times that required to solve the 512-bit RSA cryptosystem (see [8]) and was close to 400000 mips-years.

It is expected that such figures will grow slowly, unless unexpected discoveries appear in the area. From the predictions in [25], one can infer that the proposed range of parameters will presumably allow for a choice that guarantees security for the foreseeable future, at least for the next 50 years, provided the lowest values of the parameters are gradually discarded.

2.2.2 Koblitz curves

Anomalous curves are those which contain a p-torsion point other than the infinity point $\mathcal{O}$, or, equivalently, those whose Frobenius map has trace congruent to one modulo p. An anomalous elliptic curve over $GF(2^m)$ is actually defined by

$$y^2 + xy = x^3 + x^2 + 1$$

or else

$$y^2 + xy = x^3 + 1$$

The use of these curves has been suggested by Koblitz in [24]. They have complex multiplication by $K = \mathbf{Q}(\sqrt{-7})$. Their Frobenius automorphism τ, defined by

$\tau(x,y) = (x^2, y^2)$, can be used to provide an efficient computation of pairs $(k, k.G)$, as shown by work of Solinas ([48]).

In recent work (see [17, 51]), it was shown how to improve the ρ-method by a multiplicative factor $\sqrt{2}$. This takes advantage of the fact that one can simultaneously handle a point X and its opposite $-X$. Slightly better improvements can be obtained for specific curves with automorphisms, such as anomalous curves. Instead of trying to obtain collisions in the set of elliptic curve points, one can try to obtain collisions in the set of equivalence classes modulo the action of the Frobenius map. If we can achieve this, the computing time needed will be divided by something like $\sqrt{m}$, which may be significant. The technical twist that allows not to miss any collision within an equivalence class is the ability to define a canonical element in the equivalence class of a point X consisting of all $\tau^i(X)$, $i = 0, \cdots, m-1$. To define this canonical element, one can use a normal basis representation: each point of the curve is given by a sequence of $2m$ zero-one coordinates and the action of the Frobenius map is a shift of both coordinates. The canonical element corresponds to the binary sequence representing the largest integer. It is essentially unique and, in all formulas above, X_{i+1} should be replaced by the canonical element of its equivalence class and the discrete logarithm should be updated using complex multiplication as appropriate. A further improvement may use a canonical element in the set of all points of the form $\pm\tau^i(X)$. This gains another multiplicative factor $\sqrt{2}$ in terms of computing time. The resulting factor $\sqrt{2m}$ should not be overemphasized. However, the improvement definitely shows that unexpected weaknesses can follow from the use of very specific curves.

2.3 Subexponential attacks

As is well known, there are three classes of elliptic curves for which non trivial attacks have been found. They are

1. the supersingular curves

2. the anomalous curves

Supersingular curves over a field $\mathbb{F}_r$, with r a power of p, are defined by the condition that the trace of the Frobenius map is zero modulo p. For such curves, Menezes, Okamoto and Vanstone (MOV) have shown how to reduce the discrete logarithm problem to the DLP in an extension field $\mathbb{F}_{r^k}$ of $\mathbb{F}_r$, with small k. Note that, for elliptic curves over a prime field $\mathbb{Z}_p$, those curves have exactly $p+1$ elements and are specifically excluded by the key generation algorithm of [10], which performs the following check

$$p^B \neq 1 \bmod q \qquad \text{for any } 1 \leq B \leq 20$$

As already stated, anomalous curves are those which contain a p-torsion point other than $\mathcal{O}$, or, equivalently, those whose Frobenius map has trace congruent to one modulo

p. For such curves, work of Semaev [43], Rück [38], Smart [47] and Satoh-Araki [39] has shown how to solve the p-part of the DLP in polynomial time. Note that, for elliptic curves over a prime field $\mathbb{Z}_p$, those curves have exactly p elements and are specifically excluded by the key generation algorithm of [10]. Also, note that the algorithms found do not produce anything when $p = 2$. Hence, the question of the strength of the DLP in anomalous elliptic curves over fields of characteristic 2, also called Koblitz curves in section 2.2.2, remains wide open. Although, working with such curves essentially means having a single curve available for a fixed field, which may be considered highly undesirable, no subexponential algorithm is known for attacking the ECDLP.

The MOV reduction constructs an embedding from the curve into the multiplicative group of a suitable extension field of $\mathbb{F}_r$, and can be applied in a more general setting than originally envisioned by the authors. However, if the basepoint is an element of order q, q is necessarily a divisor of $r^k - 1$. Recently, Balasubramanian and Koblitz have shown in [2] that this condition was sufficient to carry the MOV reduction. The key generation algorithm specifically addresses this question. In the case of curves over $\mathbb{F}_p$, one gets that $p^k = 1 \bmod q$. From this, it follows that k is > 20, which is large enough to turn down subexponential algorithms in the extension field. In the case of curves over $\mathbb{F}_{2^m}$, there is an analogous test

$$2^{mB} \neq 1 \bmod q \qquad \text{for any } 1 \leq B \leq 20$$

with the same consequences.

Another reduction similar to the MOV reduction has appeared in the literature. It is due to Frey and Rück [16] (see also [15]) and can be stated in the more general context of jacobians on which the Tate pairing exists. Let m be an integer relatively prime to r, and let $\mu_m(\mathbb{F}_q)$ be the group of roots of unity in $\mathbb{F}_r$ whose order divides m. Assume that the Jacobian $J(\mathbb{F}_r)$ contains a point of order m. Then there is a surjective pairing

$$\varphi_m : J_m(\mathbb{F}_r) \times J(\mathbb{F}_r)/mJ(\mathbb{F}_r) \rightarrow \mu_m(\mathbb{F}_r)$$

which is computable in $\mathcal{O}(\log r)$, where $J_m(\mathbb{F}_r)$ is the group of m-torsion points. This pairing, the so-called Tate pairing, can be used to relate the discrete logarithm in the group $J_m(\mathbb{F}_r)$ to the discrete logarithm in some extension $\mathbb{F}^{\star}_{r^k}$. In the case of elliptic curves considered in the current context, the above is applicable only if the order q of the base point is a divisor of $r^k - 1$. As a consequence, the curves produced by the key generation algorithm of [10] are protected against the FR reduction, exactly due to the same argument used for the MOV reduction.

3 The security of digital signature schemes

3.1 Formal framework

In modern terms (see [20]), a digital signature scheme consists of three algorithms $(\mathcal{K}, \Sigma, V)$:

- A *key generation algorithm* $\mathcal{K}$, which, on input 1^k, where k is the security parameter, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys. Algorithm $\mathcal{K}$ is probabilistic.
- A *signing algorithm* Σ, which receives a message m and the private key sk, and outputs a signature $\sigma = \Sigma_{\mathsf{sk}}(m)$. The signing algorithm might be probabilistic.
- A *verification algorithm* V, which receives a candidate signature σ, a message m and a public key pk, and returns an answer $V_{\mathsf{pk}}(m, \sigma)$ testing whether σ is a valid signature of m with respect to pk. In general, the verification algorithm need not be probabilistic.

Attacks against signature schemes can be classified according to the goals of the adversary and to the resources that it can use. The goals are diverse:

- Disclosing the private key of the signer. It is the most drastic attack. It is termed *total break.*
- Constructing an efficient algorithm which is able to sign any message with significant probability of success. This is called *universal forgery.*
- Providing a single message/signature pair. This is called *existential forgery.*

In many cases the latter does not appear dangerous because the output message is likely to be meaningless. Nevertheless, a signature scheme, which is not existentially unforgeable, does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys or compressed data. Furthermore, it cannot formally guarantee the so-called non-repudiation property, since anyone may be able to produce a message with a valid signature.

In terms of resources, the setting can also vary. We focus on two specific attacks against signature schemes: the *no-message attacks* and the *known-message attacks.* In the first scenario, the attacker only knows the public key of the signer. In the second, the attacker has access to a list of valid message/signature pairs. Again, many subcases appear, depending on how the adversary gains knowledge. The strongest is the *adaptive chosen-message attack* (CMA), where the attacker can require the signer to sign any message of its choice, where the queries are based upon previously obtained answers. In this setting, one should point out that existential forgery becomes the ability to forge a fresh message/signature pair that has not been obtained from queries asked during the attack. There is a subtle point here: one might have the stronger rule that the attacker can only forge the signature of message, whose signature was not queried from the signing oracle. This is different only in a context where several signatures may correspond to a given message. However, the resulting security level may be strictly weaker in this case.

When designing a signature scheme, one wishes to rule out existential forgeries, even under adaptive chosen-message attacks. More formally, one requires that the success probability of any adversary $\mathcal{A}$, whose running time remains below some security bound t, is negligible, where the success probability is defined by:

$$\mathsf{Succ}^{\mathsf{cma}}(\mathcal{A}) = \Pr\left[(\mathsf{pk},\mathsf{sk}) \leftarrow \mathcal{K}(1^k), (m,\sigma) \leftarrow \mathcal{A}^{\Sigma_{\mathsf{sk}}}(\mathsf{pk}) : V_{\mathsf{pk}}(m,\sigma) = 1\right].$$

Probabilities are taken not only over the random coins of $\mathcal{A}$ but also over the random coins that $\mathcal{K}$ uses. In the above, note the superscript Σ_{sk}, indicating adaptive calls to the signing algorithm: this is consistent with the framework of relativized complexity theory, where oracle calls are allowed and, accordingly, we will use the wording *signing oracle* in this setting.

To allow for concrete estimates, we denote by $\mathsf{Succ}^{\mathsf{cma}}(t, q_s)$ the maximum success probability of an adversary whose running time is bounded by t, and who is allowed to ask at most q_s queries from the signing oracle. When $q_s = 0$, the adversary runs a no-message attack and is also called a passive adversary. We then denote its success probability by $\mathsf{Succ}^{\mathsf{nma}}(\mathcal{A})$. In other words, $\mathsf{Succ}^{\mathsf{nma}}(t) \stackrel{\mathsf{def}}{=} \mathsf{Succ}^{\mathsf{cma}}(t, 0)$.

3.2 A short history of DL-based signature schemes

3.2.1 The El Gamal scheme

The El Gamal signature scheme [13] appeared in 1985 as the first DL-based signature scheme. Rather than providing a full description of the scheme, we have depicted its components on figure 1. Note that the figure splits key generation in two steps. This stems from the fact that DL-based schemes usually set up a group $\mathcal{G}$ and a generator $\mathbf{g}$, that may be common to several users.

3.2.2 The Schnorr scheme

In 1986, Fiat and Shamir [14] introduced a new paradigm for deriving signature schemes from fair zero-knowledge identification protocols [18]. This paradigm uses hash functions to replace random queries from the verifier. In 1989, Schnorr provided a zero-knowledge identification scheme [40], together with the corresponding signature scheme [41]. Although this scheme is reminiscent of the El Gamal scheme, it uses a subgroup of order q of $\mathbb{Z}_p^\star$, where p and q are two large primes such that $q \,|\, p-1$. The Schnorr scheme appears on figure 2.

Initialization
p a large prime g a generator of $\mathbb{Z}_p^\star$
$\mathcal{K}$**: Key Generation**
private key $\quad x \in \mathbb{Z}_{p-1}$ public key $\quad y = g^x \bmod p$ $\rightarrow (y, x)$
Σ**: Signature of** m
k randomly chosen in $\mathbb{Z}_{p-1}^\star$ $r = g^k \bmod p \qquad s = (m - xr)/k \bmod p - 1$ $\rightarrow (r, s)$ a signature of m
V**: Verification of** (m, r, s)
check whether $g^m \stackrel{?}{=} y^r r^s \bmod p$ $\rightarrow$ `valid` or `invalid`

Figure 1: The El Gamal Signature Scheme.

Initialization
p, q large primes, $q \mid p - 1$ g an element of order q of $\mathbb{Z}_p^\star$
$\mathcal{K}$**: Key Generation**
private key $\quad 0 < x < q - 1$ public key $\quad y = g^{-x} \bmod p$ $\rightarrow (y, x)$
Σ**: Signature of** m
k randomly chosen $0 < k < q$ $r = g^k \bmod p \qquad e = H(m, r)$ $s = k + ex \bmod q$ $\rightarrow (e, s)$ a signature of m
V**: Verification of** (m, r, s)
check whether $\quad H(m, (g^s y^e \bmod p)) \stackrel{?}{=} e$ $\rightarrow$ `valid` or `invalid`

Figure 2: The Schnorr Signature Scheme.

```
Initialization
$\mathcal{G}$ a cyclic group of prime order $q$
$\mathbf{g}$ a generator of $\mathcal{G}$
$H : \{0,1\}^\star \to \{0,1\}^h$ a hash function
$f : \mathcal{G} \to \mathbb{Z}_q$ a reduction function

$\mathcal{K}$: Key Generation
private key    $0 < x < q$
public key     $\mathbf{y} = x \cdot \mathbf{g}$
$\to (\mathbf{y}, x)$

$\Sigma$: Signature of $m$
$k$ randomly chosen $0 < k < q$
$\mathbf{r} = k \cdot \mathbf{g}$    $r = f(\mathbf{r})$
if $r = 0$ abort and start again
$e = H(m)$    $s = k^{-1}(e + xr) \bmod q$
if $s = 0$ abort and start again
$\to (r, s)$ a signature of $m$

$V$: Verification of $(m, r, s)$
check whether $0 < r, s < q$ and $r = f(\mathbf{r}')$
where $e = H(m)$ and $\mathbf{r}' = es^{-1} \cdot \mathbf{g} + rs^{-1} \cdot \mathbf{y}$
$\to$ valid or invalid
```

Figure 3: The Generic DSA

3.2.3 DSA, ECDSA and their variants

In 1994, a digital signature standard DSA was proposed, whose flavour was a mixture of El Gamal and Schnorr. Rather than describing the original DSA [30, 32], we follow [5], and propose the description of a generic DSA (see Figure 3), which operates in any cyclic group $\mathcal{G}$ of prime order q.

In the DSA, the reduction function f takes as input an integer modulo p and outputs $f(r) = r \bmod q$. In the elliptic curve version [1, 32, 10, 11], the function is defined in a more intricate manner, which we now describe. An elliptic curve point $\mathbf{r}$ is given by two coordinates (x, y), which take values in the base field. For elliptic curves over prime fields, one simply sets $f(\mathbf{r}) = x \bmod q$. For curves over $\mathbb{F}_{2^m}$, x is a sequence of m bits and $f(\mathbf{r})$ is obtained by first turning x into an integer $< 2^m$, by a standard conversion routine.

3.3 The random oracle model

Ideally, one would like to design a signature scheme which has provable security, based on the sole assumption that the discrete logarithm or its elliptic curve variant ECDL is hard. Unfortunately, no such scheme is currently known that allows such a proof.

The next step is to hope for a proof carried in a non-standard computational model, as proposed by Bellare and Rogaway [3], following an earlier suggestion by Fiat and Shamir [14]. In this model, called the random oracle model, concrete objects such that hash functions are treated as random objects. This allows to carry through the usual reduction arguments to the context of relativized computations, where the hash function is treated as an oracle returning a random answer for each new query. A reduction still uses an adversary as a subroutine of a program that contradicts a mathematical assumption, such as the assumption that ECDL is hard However, probabilities are taken not only over coin tosses but also over the random oracle.

Of course, the significance of proofs carried in the random oracle is debatable. Firstly, hash functions are deterministic and therefore do not return random answers. Along those lines, Canetti *et al.* [7] gave an example of a signature scheme which is secure in the random oracle model, but insecure under any instantiation of the random oracle. Secondly, proofs in the random oracle model cannot easily receive a quantitative interpretation. One would like to derive concrete estimates - in terms of key sizes - from the proof: if a reduction is efficient, the security "loss" is small and the existence of an efficient adversary leads to an algorithm for solving the underlying mathematical problem, which is almost as efficient. Thus, key sizes that outreach the performances of the known algorithms to break the underlying problem, can be used for the scheme as well.

Despite these restrictions, the random oracle model has proved extremely useful to analyze many encryption and signature schemes. It clearly provides an overall guarantee that a scheme is not flawed, based on the intuition that an attacker would be forced to use the hash function in a non generic way. In the area of signature schemes based on the discrete logarithm problem, the security of the Schnorr signature scheme, based on the assumption that the hash function H behaves like a random oracle, was long considered a folklore result. A formal treatment along with concrete estimates appeared in [33, 34]. The same authors also proved the security against existential forgeries of a slight variant of the El Gamal scheme, under adaptive chosen-message attacks. The variant simply replaces the message m by $H(m, r)$, where $r = g^k \bmod p$, is computed as shown on figure 1. The proof was using the random oracle model, and was carried in the probabilistic framework developped in section 3.1. It turned out that this was an essential ingredient of the proof. Indeed, Bleichenbacher [4] showed that specific weak choices of g allowed universal forgery, both for the El Gamal scheme and for the variant. As a practical consequence, it is extremely important that users receive the guarantee that g has been chosen randomly. To offer such guarantee, standards such as [32] have developped a methodology, allowing to produce this parameter in a random and verifiable manner, again by means of hash functions.

Turning to DSA-like schemes, we mention that the generic DSA has been proven secure by techniques similar to [33, 34], under the assumption that both f and H behave like random oracles [35]. The assumptions were slightly relaxed in [5], where security was obtained, taking f as a random oracle, while only requiring that H is collision-resistant. On the other hand, no formal security proof, carried in the random oracle model, has ever been provided for either DSA or ECDSA. This is in contrast with the Korean variant of DSA, KCDSA [23], whose security has been established using random oracle techniques [5].

3.4 Generic Algorithms

Recently, several authors have proposed to use yet another model to argue in favour of the security of cryptographic schemes, that could not be tackled by the random oracle model. This is the so-called *black-box* group model, or *generic* model [42, 6, 27]. In particular, paper [6] considered the security of ECDSA in this model. Generic algorithms had been earlier introduced by Nechaev and Shoup [29, 45], to encompass group algorithms that do not exploit any special property of the encodings of group elements other than the property that each group element is encoded by a unique string. Typically, algorithms like Pollard's ρ algorithm [36] fall under the scope of this formalism while index-calculus methods do not.

Recall that any Abelian finite group Γ is isomorphic to a product of cyclic groups of the form $(\mathbb{Z}_{p^k}, +)$, where p is a prime. Such groups will be called standard groups. An encoding of a standard group Γ is an injective map from Γ into a set of bit-strings S. We give some examples: consider the multiplicative group of invertible elements modulo some prime p. This group is cyclic and isomorphic to the standard additive group $\Gamma = \mathbb{Z}_{p-1}$. Given a generator g, an encoding σ is obtained by computing the binary representation $\sigma(x)$ of $g^x \bmod p$. The same construction applies when one considers a multiplicative subgroup of prime order q. Similarly, let E be the group of points of some non-singular elliptic curve over a finite field $\mathbb{F}$, then E is either isomorphic to a (standard) cyclic group Γ or else is isomorphic to a product of two cyclic groups $\mathbb{Z}_{d_1} \times \mathbb{Z}_{d_2}$. In the first case, given a generator $\mathbf{g}$ of E, an encoding is obtained by computing $\sigma(x) = x \cdot \mathbf{g}$, where $x \cdot \mathbf{g}$ denotes the scalar multiplication of $\mathbf{g}$ by the integer x and providing coordinates for $\sigma(x)$. The same construction applies when E is replaced by one of its subgroups of prime order q. Note that the encoding set appears much larger than the group size, but compact encodings using only one coordinate and a sign bit ± 1 exist and for such encodings, the image of σ is included in the binary expansions of integers $< tq$ for some small integer t, provided that q is close enough to the size of the underlying field $\mathbb{F}$. This is exactly what is recommended for cryptographic applications [22, 10].

A *generic* algorithm $\mathcal{A}$ over a standard Abelian group Γ is a probabilistic algorithm that takes as input an *encoding list* $\mathcal{L} = \{\sigma(x_1), \cdots, \sigma(x_k)\}$, where each x_i is in Γ. While it executes, the algorithm may consult an oracle for further encodings. Oracle calls consist of triples $\{i, j, \epsilon\}$, where i and j are indices of the encoding list $\mathcal{L}$ and ϵ is $\pm$. The oracle returns the string $\sigma(x_i \pm x_j)$, according to the value of ϵ and this bit-string is appended to the list $\mathcal{L}$, unless it was already present. In other words, $\mathcal{A}$ cannot access an element of Γ directly but only through its name $\sigma(x)$ and the oracle provides names for the sum or difference of two elements addressed by their respective names. Note however that $\mathcal{A}$ may access the list $\mathcal{L}$ at any time. In many cases, $\mathcal{A}$ takes as input a pair $\{\sigma(1), \sigma(x)\}$. Probabilities related to such algorithms are computed with respect to the internal coin tosses of $\mathcal{A}$ as well as the random choices of σ and x.

In [6], the adversary is furthermore allowed to include additional elements z'_i in the encoding list $\mathcal{L}$, without calling the oracle. This is consistent with the fact that compact encodings for elliptic curves, as described above, are such that almost all bit-strings encode elliptic curve elements. However, this definitely enlarges the class of generic algorithm, compared to [29, 45]. One can keep the number of additional elements smaller than twice the number of queries, since additional elements not appearing in a further query can be deleted and since each query involves at most two additional elements.

Generic algorithms have been introduced in [29, 45], to provide lower bounds for a class of algorithms that attempt to solve the discrete logarithm problem in a generic way. The following theorem adapts the result from [45] to the extended model considered in [6].

Theorem 1 *Let Γ be a standard cyclic group of prime order q. Let $\mathcal{A}$ be a generic algorithm over Γ that makes at most n queries to the oracle. The probability that $\mathcal{A}$ returns x on input $\{\sigma(1), \sigma(x)\}$ is less than $1/q + 5 \times (n+1)^2/q$.*

Before proving the theorem, we state the following lemma, where q is still a prime number, and where an *affine* polynomial is an element P of $\mathbb{Z}_q[X_1, \ldots, X_j]$ of the form $a_0 + \sum_{i=1}^{i=j} a_i X_i$, with all its coefficients random variables.

Lemma 1 *Let P be a non-zero affine polynomial in $\mathbb{Z}_q[X_1, \ldots, X_j]$, then*

$$\Pr_{x_1, \ldots, x_j \in \mathbb{Z}_q} [P(x_1, \ldots, x_j) = 0] \leq \frac{1}{q}.$$

Proof. We write $P = a_0 + \sum_{i=1}^{i=j} a_i X_i$, and condition on each fixed sequence a_i. For such sequence, we let k be the largest index such that $a_k \neq 0$. The existence of k follows from the hypothesis that P is non-zero. Clearly,

$$p_k = \Pr_{x_1, \ldots, x_k} [P_k(x_1, \ldots, x_k) = 0]$$

$$\begin{aligned} &= \Pr[x_k = -P_{k-1}(x_1, \ldots, x_{k-1})/a_k] \\ &\leq \frac{1}{q}. \end{aligned}$$

□

Proof.(of theorem 1). Basically, we would like to identify the probabilistic space consisting of σ and x with the space $S^{n+2} \times \Gamma \times \Gamma^{2n}$, where S is the set of bit-string encodings. Given a tuple $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ in this space, z_1 and z_2 are used as $\sigma(1)$ and $\sigma(x)$, the successive z_i are used in sequence to answer the oracle queries and the $x_i \in \Gamma$ serve as pre-images of the additional elements z'_i included by the adversary into the encoding list $\mathcal{L}$. However, this interpretation may yield inconsistencies as it does not take care of possible collisions.

We give another interpretation of the encoding σ. This interpretation is based on defining from the tuple $\{z_1, \cdots, z_{n+2}\}$, a sequence of affine polynomials $F_i(X, X_1, \ldots, X_{2n})$, with coefficients modulo q, depending on the execution of $\mathcal{A}$:

- Polynomials F_1 and F_2 are set to $F_1 = 1$ and $F_2 = X$, respectively. Thus $\mathcal{L} = \{F_1, F_2\}$.
- When the adversary puts an additional k-th element z'_k in the encoding list, polynomial F_{n+k+2} is defined as X_k, and added to $\mathcal{L}$.
- At the ℓ-th query $\{i, j, \epsilon\}$, polynomial F_ℓ is defined as $F_i \pm F_j$, where the sign $\pm$ is chosen according to ϵ. If F_ℓ is already listed as a previous polynomial $F_h \in \mathcal{L}$, then F_ℓ is marked and $\mathcal{A}$ is fed with the answer corresponding to h. Otherwise, z_ℓ is returned by the oracle and F_ℓ is added to $\mathcal{L}$.

Once $\mathcal{A}$ has come to a stop, variable X is set to x, and the X_ks are set to x_k. In other words, σ is set at random, subject to the conditions $z_\ell = \sigma(F_\ell(x, x_1, \ldots, x_{2n}))$, $\ell = 1, \cdots, n+2$ and $z'_k = \sigma(x_k)$, $k = 1, \cdots, 2n$. It is easy to check that the behavior of the algorithm that is driven by the polynomials F_i is exactly similar to the behavior of the regular algorithm, granted that elements in the sequence $(z_1, \cdots, z_{n+2})$ are all distinct, and that no polynomial $F_i - F_j$ vanishes at $(x, x_1, \ldots, x_{2n})$, where i, j range over the $3n+2$ indices of polynomials in $\mathcal{L}$. We call a sequence $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ which satisfies both requirements a *safe* sequence. We bound the probability of unsafe sequences as follows, assuming n small enough, $n^2 < q$:

- Bad sequences of random elements $\{z_1, z_2, \ldots, z_{n+2}\}$, which are not all distinct appear with probability

$$\begin{aligned} p &= 1 - \prod_{k=1}^{k=n+1} \left(1 - \frac{k}{q}\right) \leq 1 - \left(1 - \sum_{k=1}^{k=n+1} \frac{k}{q}\right) \\ &\leq \frac{(n+1)(n+2)}{2q}. \end{aligned}$$

- From lemma 1, we can bound the probability that $F_i - F_j$ vanishes at $(x, x_1, \ldots, x_{2n})$ by $1/q$. Since there are at most $\binom{3n+2}{2}$ such polynomials, we infer that, once $\{z_1, \cdots, z_{n+2}\}$ have been set and are distinct, the set of $(x, x_1, \ldots, x_{2n})$ such that $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ is not safe has probability bounded by $\binom{3n+2}{2}/q = (3n+2)(3n+1)/2q$.

Finally, the probability of unsafe sequences is at most

$$\frac{(n+1)(n+2)}{2q} + \frac{(3n+2)(3n+1)}{2q} \leq \frac{5(n+1)^2}{q}.$$

As explained, an encoding σ can be defined from a safe sequence, such that:

$$\begin{aligned} \sigma(F_i(x, x_1, \ldots, x_{2n})) &= z_i, \text{ for all unmarked } F_i, \text{ and } 1 \leq i \leq n+2, \\ \sigma(x_k) &= z'_k, \text{ for } k = 1, \ldots, 2n. \end{aligned}$$

This correspondence preserves probabilities. However, it does not completely cover the sample space $\{\sigma, x\}$ since executions such that $F_i(x, x_1, \ldots, x_{2n}) = F_j(x, x_1, \ldots, x_{2n})$, for some indices i, j, such that F_i and F_j are not identical are omitted. From the above, we see that we have actually discarded at set of probability $\leq \frac{5(n+1)^2}{q}$. To conclude the proof, we simply note that the output of a computation corresponding to a safe sequence $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$ does not depend on x. Hence it is equal to x with probability $1/q$. □

4 The security of the generic DSA

In this section, we prove, in the generic model, the security of the generic DSA signature scheme proposed in section 3.2.3. We follow [6], but we adopt a different style of proof, inspired by Shoup [46]. Referring to figure 3, we clarify our use of encodings. The base point $\mathbf{g}$ of the group $\mathcal{G}$ is identified with the canonical generator 1 of $\mathbb{Z}_q$ and therefore labelled by $\sigma(1)$. Similarly, the public key $\mathbf{y}$ is labelled by $\sigma(x)$, where x is the private key. When an element $\mathbf{r}$ is requested, at signature generation, it is obtained as $\sigma(k)$, where k is randomly chosen. Finally, the reduction function f directly operates on the set of encodings S.

Contrary to the earlier approach of [42], we do not model the hash function as a random oracle. Rather, along the lines first investigated in [5], we use specific properties of the hash function, such as collision-freeness or uniformity. Similarly, we need specific properties of the reduction function. Before providing the proof, we briefly set up notations that allow to handle such properties in a quantitative manner.

4.1 A framework for the hash function and reduction function

4.1.1 One-Wayness

Let H be a hash function $H : \{0,1\}^* \to \{0,1\}^h$. An (ε, t)-inverter for H is an algorithm able to invert H with probability greater than ε, and with running time bounded by t. Probabilities are taken over a random element $b \in_R \{0,1\}^h$:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[b \in_R \{0,1\}^h, a \leftarrow \mathcal{A}(b) : H(a) = b] \geq \varepsilon.$$

Denote by $\mathsf{Succ}^{\mathsf{ow}}(H,t)$ the maximal success probability for inverting H over all adversaries whose running time is bounded by t. Hash function H is (ε, t)-one-way if $\mathsf{Succ}^{\mathsf{ow}}(H,t)$ is upper-bounded by ε.

Let q be a prime number. Identifying elements of $\{0,1\}^h$ with integers $< 2^h$, one can define $H' : \{0,1\}^* \to \{0, \ldots q-1\}$ by $H'(m) \overset{\mathsf{def}}{=} H(m) \bmod q$. The definition holds whether q is $< 2^h$ or $\geq 2^h$.

When q is $<< 2^h$, then inverting H' is easier than inverting H. Assume now that q is such that $\frac{2^h}{\theta} < q \leq 2^h$, for some small constant $\theta > 1$. Using modular reduction, an inverter for H', with success probability ε' easily translates into an inverter for H, with success probability $\varepsilon \geq \frac{\varepsilon'}{\theta}$. A similar argument holds if $2^h < q < \theta 2^h$. Thus, provided 2^h and q are of the same order of magnitude, inverting H and H' is equally difficult. On the other hand, when $2^h << q$, then any inverter for H', has success probability $\leq \frac{2^h}{q}$.

4.1.2 Uniformity

Let H be as above and $\mathcal{M}$ be a distribution on a set of messages. Hash function H is (δ, t)-*uniform* on $\mathcal{M}$, if the distribution $\{H(\mathcal{M})\}$ is δ-indistinguishable from the uniform distribution. More precisely, no distinguisher $\mathcal{D}$, with running time bounded by t, can get an advantage greater than δ, where the advantage of $\mathcal{D}$ is the difference between the probabilities that $\mathcal{D}$ outputs one, with inputs taken from each distribution.

4.1.3 Collision-Resistance

Let H be as above. A (γ, t)-collision-finder for H is an algorithm $\mathcal{A}$, running in time bounded by t, and able to find a collision with probability greater than γ:

$$\mathsf{Succ}^{\mathsf{col}}(\mathcal{A}) = \Pr[(a_0, a_1) \leftarrow \mathcal{A} : H(a_0) = H(a_1)] \geq \gamma.$$

Denote by $\mathsf{Succ}^{\mathsf{col}}(H,t)$ the maximal success probability for finding a collision, taken over all adversaries whose running time is bounded by t. Hash function H is (γ, t)-collision-resistant if $\mathsf{Succ}^{\mathsf{col}}(H,t)$ is upper-bounded by γ.

4.1.4 Almost-Invertibility

Let f be a reduction function $f : S \to \mathbb{Z}_q$. An almost-inverse g of f is a probabilistic algorithm g, possibly outputting Fail, such that

$$(i) \quad \Pr_{b \in_R \mathbb{Z}_q} [g(b) \in S \wedge f(g(b)) = b] \geq 1/3$$

Function f is (δ, t)-*almost-invertible*, with almost-inverse g, if furthermore:

$$(ii) \quad \mathcal{D}_g \approx_\delta \mathcal{U}, \text{ where } \left\{ \begin{array}{rcl} \mathcal{D}_g & = & \{g(b) \,|\, b \in_R \mathbb{Z}_q \wedge g(b) \in S\} \\ \mathcal{U} & = & \{a \,|\, a \in_R S\}. \end{array} \right.$$

In the second item, notation $\mathcal{D}_g \approx_\delta \mathcal{U}$ means that no distinguisher with running time bounded by t can get an advantage greater than δ.

4.2 Security against passive adversaries

In this section we prove the security of the generic DSA against passive adversaries, *i.e.* adversaries which try to forge a message/signature pair without querying the signing oracle.

Theorem 2 *Let Γ be a standard cyclic group of prime order q. Let S be a set of bit-string encodings. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function and $f : S \to \mathbb{Z}_q$ be a reduction function with almost-inverse g. Let $\mathcal{A}$ be a generic algorithm over Γ, that makes at most n queries to the group-oracle. Assume that $\mathcal{A}$, on input $\{\sigma(1), \sigma(x)\}$, returns a message m and a valid generic DSA signature of m, with probability $\varepsilon = \mathsf{Succ}^{\mathsf{nma}}(\mathcal{A})$, within running time t. Then there exist adversaries $\mathcal{B}_H$, $\mathcal{B}_g$, operating within time bound t', and such that $\mathcal{B}_H$ is attempting to invert $H' = H \bmod q$ with success probability ε_H, and $\mathcal{B}_g$ is playing a distinguishing game for g, with advantage ε_g, where*

$$\begin{array}{rcl} \varepsilon & \leq & 3n\varepsilon_H + 3n\varepsilon_g + \dfrac{5(n+1)^2}{q}, \\ t' & \leq & t + \tau. \end{array}$$

with τ the running time of g.

In order to prove this result, we define a sequence Game_1, Game_2, etc of modified attack games starting from the actual game Game_0. Each of the games operates on the same underlying probability space: the public and private key of the scheme, the coin tosses of the adversary $\mathcal{A}$, the random encoding σ. Only the rules defining how the view is computed differ from game to game. To go from one game to another, we repeatedly use the following lemma from [46]:

Lemma 2 *Let* E_1*,* E_2 *and* F *be events defined on a probabilistic space*

$$\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] = \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] \implies |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| \leq \Pr[\mathsf{F}].$$

Proof. The proof follows from easy computations:

$$\begin{aligned} |\Pr[\mathsf{E}_1] - \Pr[\mathsf{E}_2]| &= |\Pr[\mathsf{E}_1 \wedge \neg\mathsf{F}] + \Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \neg\mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| \\ &= |\Pr[\mathsf{E}_1 \wedge \mathsf{F}] - \Pr[\mathsf{E}_2 \wedge \mathsf{F}]| = |\Pr[\mathsf{E}_1 \,|\, \mathsf{F}] \cdot \Pr[\mathsf{F}] - \Pr[\mathsf{E}_2 \,|\, \mathsf{F}] \cdot \Pr[\mathsf{F}]| \\ &\leq |\Pr[\mathsf{E}_1 \,|\, \mathsf{F}] - \Pr[\mathsf{E}_2 \,|\, \mathsf{F}]| \cdot \Pr[\mathsf{F}] \leq \Pr[\mathsf{F}] \end{aligned}$$

□

Proof.(of theorem 2). We use the probabilistic model developed in section 3.4. Let $\mathcal{A}$ be a generic attacker able to forge a pair consisting of a message m and a valid signature (r, s). We assume that, once these outputs have been issued, $\mathcal{A}$ goes on checking the signature: this means requesting the encoding of $es^{-1} + xrs^{-1} \bmod q$, where $e = H(m)$, and checking that its image under f is r. The request can be performed by mimicking the usual double-and-add algorithm, calling the generic encoding at each group operation. However, to keep things simple, we ignore the number of additional requests to the group-oracle. The extra check allows to estimate the probability that the output signature is valid. As explained, we start with the game coming from the actual attack, and modify it step by step until we reach a final game, whose success probability has an upper-bound obviously related to inverting the hash function.

Game_0**:** An encoding σ is chosen and a key pair $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary $\mathcal{A}$ is fed with pk and, querying the generic encoding, outputs a message m and a signature (r, s). We denote by S_0 the event $V_{\mathsf{pk}}(M, (r, s)) = 1$ and use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1**:** We slightly modify this game, by using the interpretation of the encoding proposed in section 3.4: this uses a random sequence $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$. As shown in section 3.4, the new game only differs from the old on unsafe sequences:

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{5(n+1)^2}{q}.$$

Game_2**:** In this game, we choose at random an index κ between 1 and $n + 2$. We know that F_{n+2} is $es^{-1} + rs^{-1}X$, since it appears as the final step of the verifying algorithm. We then discard executions where F_κ is not the first occurrence of F_{n+2}. Since the additional random value κ is chosen independently of the execution of Game_1, and since it hits an occurrence of F_{n+2} with probability $\geq 1/n$, we get

$$\Pr[S_2] \geq Pr[S_1] \times \frac{1}{n}.$$

Game$_3$: In this game, we make an additional test on κ. Let $F_\kappa = b_\kappa X + a_\kappa$. We pick at random $\tilde{e} \in_R \mathbb{Z}_q$, and compute $c_\kappa \leftarrow g(b_\kappa a_\kappa^{-1}\tilde{e} \bmod q)$, where g is the probabilistic inverse of f. If the computation of g returns Fail, we abort the game. We let GBad the probability that this happens. From the randomness of $\tilde{e}$ and the fact that g is an almost-inverse of f, we get $\Pr[\neg \mathsf{GBad}] \geq 1/3$. Now, the new experiment is independent from the execution of Game$_2$. Thus:

$$\begin{aligned} \Pr[S_3] &= \Pr[S_2] \times \Pr[\neg \mathsf{GBad}] \\ &\geq \Pr[S_2] \times \frac{1}{3}. \end{aligned}$$

Game$_4$: Here, we further modify the previous game by letting c_κ replace z_κ. Since $\tilde{e}$ is uniformly distributed, the input to g is uniformly distributed as well. Using the *almost-invertibility* of f, we bound the difference between the success probabilities of the two games by δ_g: $|\Pr[S_4] - \Pr[S_3]| \leq \delta_g$.

We finally upper-bound $\Pr[S_4]$. We observe that that, while checking the signature, the final request of the adversary is the encoding of $es^{-1} + xrs^{-1} \bmod q$. The answer in Game$_4$ is $c_\kappa = g(b_\kappa a_\kappa^{-1}\tilde{e} \bmod q)$, and the image r' under f of this answer is $b_\kappa a_\kappa^{-1}\tilde{e} \bmod q$. Since a_κ is $es^{-1} \bmod q$, and b_κ is $rs^{-1} \bmod q$, we get that $r' = r$ if and only if $\tilde{e} = e \bmod q$, which means that a pre-image of $\tilde{e}$ has been obtained. Therefore, $\Pr[S_4] \leq \varepsilon_H = \mathsf{Succ}^{\mathsf{ow}}(H', t')$, where $H'(m) \overset{\mathsf{def}}{=} H(m) \bmod q$. Summing up inequalities, we get:

$$\begin{aligned} \Pr[S_0] &\leq \Pr[S_1] + 5 \times (n+1)^2/q \leq n \Pr[S_2] + 5 \times (n+1)^2/q \\ &\leq 3n \times \Pr[S_3] + 5 \times (n+1)^2/q \\ &\leq 3n \times (\Pr[S_4] + \delta_g) + 5 \times (n+1)^2/q \\ &\leq 3n\varepsilon_H + 3n\delta_g + 5 \times (n+1)^2/q. \end{aligned}$$

□

4.3 Active Adversaries

In this section we extend the previous result to active adversaries, which have access to the signing oracle. First of all, we clarify our notion of existential forgery: a forgery, that provides a second signature of a message for which the adversary has already obtained one from the signing oracle, is duly accepted. This means that we cover the stronger security notion, based on a relaxed definition of existential forgery.

A couple of lemmas will be needed. We first show how one can perfectly simulate the distribution of valid signatures. We define a simulator $\mathcal{S}$. The simulator, picks elements $u \in_R S$, and $s \in_R \mathbb{Z}_q$, and outputs the pair (r, s), with $r = f(u)$.

Lemma 3 *For any message m, the output distribution of $\mathcal{S}$ is perfectly indistinguishable from the output distribution of $\Sigma_{\mathsf{sk}}(m)$.*

Proof. We have to compare two distributions: the original distribution $\mathcal{D}_0$ generated by Σ on a fixed message m, and the distribution $\mathcal{D}_1$ produced by $\mathcal{S}$.

$$\begin{aligned} \mathcal{D}_0 &= \{(r,s) \,|\, x \in_R \mathbb{Z}_q; k \in_R \mathbb{Z}_q^\star; \sigma \in_R \mathbb{Z}_q^S; r = f(\sigma(k)); s = k^{-1}(H(m) + xr) \bmod q\} \\ \mathcal{D}_1 &= \{(r,s) \,|\, u \in_R S; s \in_R \mathbb{Z}_q; r = f(u)\} \end{aligned}$$

That they coincide is shown by the following equalities:

$$\begin{aligned} \mathcal{D}_0 &= \{(r,s) \,|\, x \in_R \mathbb{Z}_q; k \in_R \mathbb{Z}_q^\star; \sigma \in_R \mathbb{Z}_q^S; r = f(\sigma(k)); s = k^{-1}(H(m) + xr) \bmod q\} \\ &= \{(r,s) \,|\, x \in_R \mathbb{Z}_q; k \in_R \mathbb{Z}_q^\star; u \in_R \mathbb{Z}_q; r = f(u); s = k^{-1}(H(m) + xr) \bmod q\} \\ &= \{(r,s) \,|\, s \in_R \mathbb{Z}_q; u \in_R \mathbb{Z}_q; r = f(u)\} = \mathcal{D}_1. \end{aligned}$$

□

We now state an easy lemma from elementary probability theory.

Lemma 4 *Let $\mathbf{S}$ be a binomial distribution, which is the sum of $k = 5 \ln n$ Bernoulli trials with probability for success $\geq 1/3$. Then, the probability that $\mathbf{S} = 0$ is at most $\frac{1}{n^2}$.*

Proof. The probability that $\mathbf{S} = 0$ is upper-bounded by

$$\left(\frac{2}{3}\right)^{5 \ln n} = \exp\left(-5 \ln n \ln(3/2)\right) < \exp\left(-2 \ln n\right) = \frac{1}{n^2}.$$

□

Theorem 3 *Let Γ be a standard cyclic group of prime order q. Let S be a set of bit-string encodings. Let $H : \{0,1\}^* \to \{0,1\}^h$ be a hash function and $f : S \to \mathbb{Z}_q$ be a reduction function with almost-inverse g. Let $\mathcal{A}$ be a generic algorithm over Γ, that makes at most q_s queries to the signing oracle and n queries to the group-oracle, respectively. Assume that $\mathcal{A}$, on input $\{\sigma(1), \sigma(x)\}$, returns a message m and a valid generic DSA signature (r,s) of m, with probability $\varepsilon = \mathsf{Succ}^{\mathsf{cma}}(\mathcal{A})$, within running time t. Then there exist adversaries $\mathcal{B}_H$, $\mathcal{C}_H$, $\mathcal{D}_g$, operating within time bound t', and such that $\mathcal{B}_H$ is attempting to invert $H' = H \bmod q$ with success probability ε_H, $\mathcal{C}_H$ is attempting to find collisions for $H' = H \bmod q$ with success probability γ_H, and $\mathcal{D}_g$ is playing a distinguishing game for g, with advantage δ_g, where*

$$\begin{aligned} \varepsilon &\leq 2\gamma_H + 2n(\delta_g + \varepsilon_H) + \frac{5(n+1)(n+q_s+1)}{q}, \\ t' &\leq t + n \times (5\tau_g \ln n + \tau_H), \end{aligned}$$

with τ_g the running time of g and τ_H the running time for H.

Proof. Let $\mathcal{A}$ be a generic attacker able to forge a pair consisting of a message m and a valid signature (r, s). As for the passive case, we assume that, once these outputs have been issued, $\mathcal{A}$ goes on checking the signature by requesting the encoding of $es^{-1} + xrs^{-1} \bmod q$, where $e = H(m)$, and checking that its image under f is r. We assume furthermore, that, after each query m_j to the signing oracle, the adversary immediately performs a similar request to check the validity of the answer. To keep things simple, we do not perform any book-keeping of the additional requests and keep n to denote the overall number of queries to the group oracle. We now play games as before:

Game_0: An encoding σ is chosen and a key pair $(\mathsf{pk}, \mathsf{sk})$ is generated using $\mathcal{K}(1^k)$. Adversary $\mathcal{A}$ is fed with pk and, querying the generic encoding and the signing oracle, outputs a message m and a signature (r, s). We denote by S_0 the event $V_{\mathsf{pk}}(M, (r, s)) = 1$ and use a similar notation S_i in any Game_i below. By definition, we have $\Pr[S_0] = \varepsilon$.

Game_1: We slightly modify this game, by using the interpretation of the encoding proposed in section 3.4: this uses a random sequence $\{z_1, \cdots, z_{n+2}, x, x_1, \ldots, x_{2n}\}$. As shown in section 3.4, the new game only differs from the old on unsafe sequences:

$$|\Pr[S_1] - \Pr[S_0]| \leq \frac{5(n+1)^2}{q}.$$

Game_2: In this game, we perform additional random tests, without modifying the simulation of the generic oracle: a test is performed at each index ℓ, such that the corresponding affine polynomial appears for the first time (or is unmarked following the terminology of section 3.4). Let $F_\ell = b_\ell X + a_\ell$. We pick at random $\tilde{e_\ell} \in_R \mathbb{Z}_q$, and compute $c_\ell \leftarrow g(b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q)$ until the computation of g returns an answer different from Fail. However, we stop and abort the game after $5 \ln n$ trials. This game differs from the previous one if c_ℓ remains undefined after $5 \ln n$ attempts. Since $\tilde{e_\ell}$ is uniformly distributed, and since the successive trials are mutually independent, we may use lemma 4 and bound the corresponding probability by $\frac{1}{n^2}$. This provides the overall bound $1/n$, when ℓ varies. Taking into account the fact that the experiments are independent from the execution of Game_1, we get

$$\Pr[S_2] \geq (1 - \frac{1}{n}) \Pr[S_1].$$

Game_3: Here, we further modify the previous game by letting c_ℓ replace z_ℓ, for each index ℓ such that F_ℓ is unmarked. Note that we have $f(z_\ell) = b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q$. Since the $\tilde{e_\ell}$s are uniformly distributed, the inputs to g are uniformly distributed as well. Applying the so-called *hybrid* technique, which amounts to using n

times the *almost-invertibility* of g, we bound the difference between the success probabilities of the two games by $n\delta_g$: $|\Pr[S_4] - \Pr[S_3]| \le n\delta_g$.

Game$_4$**:** In this game, we simulate the signing oracle. For any query m_j to the signing oracle, one computes $e_j = H(m_j)$, and issues a random signature (r_j, s_j), using the simulation of lemma 3. Recall that the simulation picks s_j at random and computes r_j as $f(u_j)$, where u_j is randomly drawn from S. By lemma 3, this simulation is perfect. Observe that, while checking the signature, the adversary requests, at some later time, the encoding of $e_j {s_j}^{-1} + x r_j {s_j}^{-1} \bmod q$. We let ℓ the first index corresponding to such query, $F_\ell = b_\ell X + a_\ell$. We modify z_ℓ, replacing its earlier value by u_j and define $\tilde{e_\ell}$ as $e_j = H(m_j)$. Observe that we still have $f(z_\ell) = b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q$. This game only differs from the previous one if polynomial $e_j {s_j}^{-1} + X r_j {s_j}^{-1}$ collides with a previous one. Due to the randomness of s_j, we obtain:

$$|\Pr[S_4] - \Pr[S_3]| \le \frac{nq_s}{q}.$$

We note that the final simulation runs in time $t' \le t + n \times (5\tau_g \ln n + \tau_H)$ and we finally upper-bound $\Pr[S_4]$. We observe that, while checking the signature, the final request of the adversary, with index $n+2$, is the encoding of $es^{-1} + xrs^{-1} \bmod q$, where $e = H(m)$. We let ℓ be the first occurrence of F_{n+2}. If the signature is valid, the following equalities hold:

$$\begin{array}{rcl} es^{-1} & = & a_\ell \bmod q \\ rs^{-1} & = & b_\ell \bmod q \\ f(z_\ell) & = & b_\ell a_\ell^{-1} \tilde{e_\ell} \bmod q \\ r & = & f(z_\ell) \end{array}$$

From these equalities, it easily follows that $r = f(z_\ell) = re^{-1}\tilde{e_\ell} \bmod q$, which in turn implies $e = \tilde{e_\ell} \bmod q$. We distinguish two cases:

- If z_ℓ has been created according to the rule of Game$_3$, then a pre-image m of some randomly chosen element $\tilde{e_\ell}$ among the n possible ones has been found.

- If z_ℓ has been created according to the rule of Game$_4$, then $\tilde{e_\ell}$ is the image under H of a message m_j queried from the signing oracle. Furthermore, we have:

$$\begin{array}{rcl} e_j s_j^{-1} & = & a_\ell \bmod q \\ r_j s_j^{-1} & = & b_\ell \bmod q \end{array}$$

  Comparing to the above equalities, we get that $s = s_j \bmod q$ and $r = r_j \bmod q$. Note that m_j cannot be equal to m, since otherwise the output forged signature would coincide with an earlier signature (r_j, s_j) of the same message m. Thus, a collision has been found for H', where $H'(m) \overset{\text{def}}{=} H(m) \bmod q$.

The probability that an algorithm running in time t' finds a preimage under H' of an element among n is at most $n\varepsilon_H$. From this, we obtain that: $\Pr[S_4] \leq n\varepsilon_H + \gamma_H$.

Summing up inequalities, we get:

$$\begin{aligned}
\Pr[S_0] &\leq \Pr[S_1] + 5 \times \frac{(n+1)^2}{q} \\
&\leq \left(1 - \frac{1}{n}\right)^{-1} \Pr[S_2] + 5 \times \frac{(n+1)^2}{q} \leq 2\Pr[S_2] + 5 \times (n+1)^2/q \\
&\leq 2 \times (\Pr[S_3] + n\delta_g) + 5 \times (n+1)^2/q \\
&\leq 2\Pr[S_4] + 2n\delta_g + +2\frac{nq_s}{q} + 5 \times (n+1)^2/q \\
&\leq 2\gamma_H + 2n(\delta_g + \varepsilon_H) + 5 \times (n+1+q_s)(n+1)/q.
\end{aligned}$$

□

5 The practical security of ECDSA

In this section, we analyze the security of the ECDSA signature scheme in view of the previous proofs. This involves a discussion on the significance of the generic model, both as a general paradigm, and in connection with its use to argue in favour of the security of ECDSA.

5.1 The generic model as a security paradigm

As noted in section 3.3, the significance of proofs carried in the random oracle is a matter of controversy, and it is further debatable whether or not security estimates obtained in the random oracle model can be used to derive key sizes. However, it is generally acknowledged that a security proof in the random oracle model provides an overall guarantee that a scheme is not flawed, based on the intuition that an attacker would be forced to use the hash function in a non generic way.

The generic model was proposed to provide lower bounds for a class of algorithms that attempt to solve the discrete logarithm problem in a generic way. This gives a restricted, albeit meaningful indication of the hardness of the discrete logarithms in general groups. Of course, the idea of further using the generic model to argue in favour of the security of specific schemes is appealing. However, we find it rather unconvincing, at a methodological level, for reasons that we now develop.

Firstly, it departs from the usual approach of provable security, which reduces a cryptographic scheme to a hard problem. Proofs in the generic model are absolute, as seen from the statements of theorems 2 and 3. Accordingly, they do not measure the loss that a cryptographic construct might bring *per se*.

Secondly, hash functions are meant to be random. They are crafted so that statistical attacks of the type considered in conventional cryptanalysis cannot be mounted. On the other hand, groups used in public key cryptography, have an underlying algebraic structure, where randomness does not play any role.

5.2 The case of ECDSA

In [6], it is argued that theorems 2 and 3 are meaningful in the context of ECDSA, while they are not in the context of DSA. We claim this is simply wrong. In order to support our claim, we review the meaning of each of the hypotheses on which the theorems rely.

5.2.1 The hash function and reduction function

Theorems 2 and 3 require the hash function H' to be sufficiently one-way and collision resistant, where $H'(m) \overset{\mathsf{def}}{=} H(m) \bmod q$. Currently, the only supported hash funstion H for ECDSA is SHA-1. This function has been designed to be one-way and collision resistant. When the size q of the group is much smaller than 2^h, where $h = 160$ is the bit-size of outputs of SHA-1, then it becomes easier to invert H' or create collisions. When q is of the same order of magnitude as 2^h, then inverting H' and inverting H is essentially equivalent and the same is true for collision finding. Finally, when q is much larger than 2^h, inverting H' is hard anyhow, whereas collision finding remains at the same computational level of hardness. In conclusion, it makes sense to assume that H' is one-way and collision resistant, as soon as q is at least of the order 2^h. We note that, contrary to [6], we have avoided to use the uniformity of H. We believe this is a better approach.

The reduction function for ECDSA [1], takes the first coordinate of a curve element and reduces it modulo q: $\mathbf{r}$, $f(\mathbf{r}) = x_{\mathbf{r}} \bmod q$. In case of binary fields, one first has to turn $x_{\mathbf{r}}$ into an integer $< 2^m$, by a standard conversion routine. Note that, when x is given in the base field, either it is not the first coordinate of a curve element, or else two points of the curve have first coordinate x. Those two points $\mathbf{r}_1$ and $\mathbf{r}_2$ are symmetric: $\mathbf{r}_1 = -\mathbf{r}_2$. Furthermore, the second coordinate of each of the two points is easily computed. This allows to sample the curve by defining a probabilistic inverse g of f at b as follows:

1. pick at random x in the base field such that $x = b \bmod q$,
2. output at random one of the two curve elements with first coordinate x or Fail if there is none.

By Hasse's Theorem, the number of elements of E is

$$\#E = p + 1 + t, \text{ where } |\, t \,| \leq 2\sqrt{p}.$$

Therefore, the probability that g succeeds is lower-bounded by $\frac{p+1-2\sqrt{p}}{p}$, which is $\geq 1/3$. This is in the setting of prime fields, but a similar result holds for binary fields. In conclusion, the assumption on the reduction function makes sense.

5.3 The encoding function

We claim that the encoding function cannot be viewed as generic. To support the claim, we observe that ECDSA is not immune to existential forgery in the stronger security model that we have considered. Indeed, from a signature (r, s) of a message m, one can derive a second one, namely $(r, -s)$. Referring to figure 3, we see that the values of $\mathbf{r}'$ that appear in the verification of both signatures are symmetric, so that their image by f is the same. If the generic model was an appropriate framework, then this would contradict theorem 3. Of course, it is presumably possible to define a notion of *symmetric generic encoding*, where encodings of $\pm\mathbf{r}$ are indentical and to search for a security result of some form, in the symmetric generic model. This looks quite twisted.

6 Conclusion

We have investigated the security of the ECDSA primitive and its relation to the elliptic curve discrete logarithm problem. Based on our analysis, we believe that the range of parameters offered by the standard can provide security for a number of years, provided the lowest figures are gradually discarded, taking into account the progress of computing power. The use of Koblitz curves, albeit suspicious, does not provide any significant shortcut for the adversary.

We also believe that the security proof that the generic DSA withstands existential forgery against adaptive chosen-message attacks, is mathematically sound. We have indeed provided a security proof in Shoup's style [46], different from the proof earlier published in [6]. However, we have doubts on the actual significance of this proof. Furthermore, we have shown that ECDSA does not withstand chosen-message attacks, where forgeries that output a second signature of a message previously signed are allowed. The generic DSA provably withstands such forgeries. This seems to indicate that the generic model is not appropriate. In conclusion, we regret that we cannot recommend ECDSA without reservations.

References

[1] American National Standards Institute. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm. ANSI X9.62-1998, January 1999.

[2] R. Balasubramanian and N. Koblitz. The improbability than an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm, *J. Cryptology*, 111, 1998, 141–145.

[3] M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73, ACM Press, New York, 1993.

[4] D. Bleichenbacher. Generating El Gamal Signatures without Knowing the Secret Key. In *Eurocrypt '96*, LNCS 1070, pages 10–18, Springer-Verlag, Berlin, 1996.

[5] E. Brickell, D. Pointcheval, S. Vaudenay, and M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *PKC '2000*, LNCS 1751, pages 276–292, Springer-Verlag, Berlin, 2000.

[6] D. R. L. Brown. The Exact Security of ECDSA, January 2001. Available from `http://grouper.ieee.org/groups/1363/`.

[7] 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.

[8] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. C. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, P. Zimmermann. Factorization of a 512-Bit RSA Modulus. Eurocrypt'2000, LNCS 1807,(2000), 1–18, Springer-Verlag, Berlin, 2000.

[9] Certicom. Information on the Certicom ECC challenge, `http://www.certicom.com/research/ecc_challenge.html`

[10] Certicom. Standards for efficient cryptography, SEC1: elliptic curve cryptography, sept, 20, 2000.

[11] Certicom. Standards for efficient cryptography, SEC2: Recommended elliptic curve parameters, sept, 20, 2000.

[12] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

[13] 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.

[14] 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.

[15] G. Frey, M. Müller, and H. G. Rück. The Tate-Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Transactions on Information Theory*, 45:1717–1719, 1999.

[16] G. Frey and H. G. Rück. A Remark Concerning m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves. *Mathematics of Computation*, 62:865–874, 1994.

[17] R. Gallant, R. Lambert and S.A. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous elliptic curves. *Mathematics of Computation*, 69, (2000), 1699–1705.

[18] S. Goldwasser, S. Micali, and C. Rackoff. The Knowledge Complexity of Interactive Proof Systems. In *Proc. of the 17th STOC*, pages 291–304, ACM Press, New York, 1985.

[19] 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.

[20] 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.

[21] R. Harley, D. Doligez, D. de Rauglaudre, X. Leroy. Elliptic Curve Discrete Logarithms: ECC2K-108,
`http://cristal.inria.fr/~harley/ecdl7/`

[22] IEEE P1363. Standard Specifications for Public Key Cryptography, August 1998. Available from `http://grouper.ieee.org/groups/1363/`.

[23] KCDSA Task Force Team. The Korean Certificate-based Digital Signature Algorithm. Submission to IEEE P1363a, August 1998.
Available from `http://grouper.ieee.org/groups/1363/`.

[24] N. Koblitz. CM-curve with good cryptographic properties. In *Crypto '92*,LNCS 576, pages 279–287, Springer-Verlag, Berlin, 1993.

[25] A.K. Lenstra and E. Verheul. Selecting cryptographic key sizes. PKC'2000, LNCS 1751,(2000), 446–465, Springer-Verlag, Berlin, 2000.

[26] J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1. In *Crypto '2001*, LNCS 2139, pages 230–238, Springer-Verlag, Berlin, 2001.

[27] D. Naccache, D. Pointcheval, and J. Stern. Twin Signatures: an Alternative to the Hash-and-Sign Paradigm. In *Proc. of the 8th CCS*, ACM Press, New York, 2001.

[28] D. Naccache and J. Stern. Signing on a Postcard. In *Financial Cryptography '2000*, LNCS. Springer-Verlag, Berlin, 2000.

[29] V. I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[30] NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186, November 1994.

[31] NIST. Secure Hash Standard (SHS). Federal Information Processing Standards PUBlication 180–1, April 1995.

[32] NIST. Digital Signature Standard (DSS). Federal Information Processing Standards PUBlication 186–2, January 2000.

[33] D. Pointcheval and J. Stern. Security Proofs for Signature Schemes. In *Eurocrypt '96*, LNCS 1070, pages 387–398, Springer-Verlag, Berlin, 1996.

[34] D. Pointcheval and J. Stern. Security Arguments for Digital Signatures and Blind Signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[35] D. Pointcheval and S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical Report LIENS-96-17, LIENS, October 1996. Available from `http://www.di.ens.fr/users/pointche/pub.html`.

[36] J. M. Pollard. Monte Carlo Methods for Index Computation (mod p). *Mathematics of Computation*, 32(143):918–924, July 1978.

[37] 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.

[38] H.G. Rück. On the discrete logarithm in the divisor class group of curves. Preprint, 1997.

[39] T. Satoh, K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves, 1997, to appear in Commentarii Math. Univ. St Pauli.

[40] C. P. Schnorr. Efficient Identification and Signatures for Smart Cards. In *Crypto '89*, LNCS 435, pages 235–251, Springer-Verlag, Berlin, 1990.

[41] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[42] C. P. Schnorr and M. Jakobsson. Security of Signed ElGamal Encryption. In *Asiacrypt '2000*, LNCS 1976, pages 458–469, Springer-Verlag, Berlin, 2000.

[43] I.A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve of characteristic p. *Math. Comp.*, 67, 1998, 353–356.

[44] D. Shanks. Class Number, a Theory of Factorization, and Genera. In *Proceedings of the Symposium on Pure Mathematics*, volume 20, pages 415–440. AMS, 1971.

[45] V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Eurocrypt '97*, LNCS 1233, pages 256–266, Springer-Verlag, Berlin, 1997.

[46] V. Shoup. OAEP Reconsidered. In *Crypto '2001*, LNCS 2139, pages 239–259, Springer-Verlag, Berlin, 2001. Also appeared in the Cryptology ePrint Archive 2000/060. November 2000. Available from `http://eprint.iacr.org/`.

[47] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12, 1999, 141–151.

[48] J.A. Solinas. An improved algorithm for arithmetic on a family of elliptic curves. In *Crypto '97*, LNCS 1294, pages 357–371, Springer-Verlag, Berlin, 1997.

[49] P.C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications, *J. Cryptology*, 12, 1999, 1–28.

[50] S. Vaudenay. Hidden Collisions on DSS. In *Crypto '96*, LNCS 1109, pages 83–88, Springer-Verlag, Berlin, 1996.

[51] M. J. Wiener and R.J. Zuccherato. Fast attacks on elliptic curve cryptosystems, SAC'98, LNCS 1556, 190–200, Springer-Verlag, Berlin, 1999.