Evaluation Report on ECDSA

Serge Vaudenay

IPA Work Delivery

Abstract. Following the contract between IPA and SV signed on 18.10.2001, this report presents security evaluation on ECDSA:
- the scheme and primitive of ECDSA
- the random generator given in FIPS 186-2 Appendix 3
- the scheme and primitive of ECDSA
- the provable security of ECDSA
- the validity of Koblitz curves

Let us first summarize the ECDSA as presented in ANSI X9.62 [2].

Public parameters: finite field $\mathbf{F}_q$, an elliptic curve C over $\mathbf{F}_q$ where q is either prime or a power of 2, a point $G \in C$ of prime order $n > 2^{160}$
Secret key: integer d in $[1, n-1]$
Public key: $Q = dG$
Signature generation for M: generate $k \in [1, n-1]$ and compute

$$\begin{aligned} (x_1, y_1) &= kG \\ r &= \overline{x_1} \bmod n \\ s &= \frac{\text{SHA-1}(M) + dr}{k} \bmod n \end{aligned}$$

the signature is (r, s)
Signature (r', s') verification for M: check that r' and s' are in $[1, q-1]$ and that $r' = \overline{x_1} \bmod n$ for $(x_1, y_1) = u_1 G + u_2 Q$, $u_1 = \frac{\text{SHA-1}(M)}{s'} \bmod n$, and $u_2 = \frac{r'}{s'} \bmod n$

Here $\overline{x_1}$ is simply a way of converting a field element into an integer and SHA-1 is a hash function specified in FIPS 180-1 [3].

The signature is a pair of integers. The public key is a point on a curve. So ANSI X9.62 [2] needs to define a standard way for representing an integer, a point, and therefore a field element. In addition we need a standard way to represent the public parameters: the field representation, the curve definition, ... ANSI X9.62 [2] extensively defines all this.

1 Background

A long history of digital signature schemes was initiated by ElGamal in his PhD in 1984 [17–19]. It included the Schnorr signature [30, 31] and DSA [4]. The use of elliptic curve in cryptography was first proposed by Koblitz and Miller. This is motivated by the belief that discrete logarithm is harder in elliptic curves with same parameter sizes. Based on the impulse of the Certicom company and the SECG group, DSA was transformed into ECDSA in order to implement elliptic curves features.

2 Necessary Security Assumptions

Obviously, **ability to compute discrete logarithms** in the subgroup of spanned by G is a potential attack against ECDSA since we can recover d. A simple attack which uses Shanks baby step giant step algorithm works within **a complexity of** $\sqrt{n}$.

Inverting SHA-1 is another potential attack. If we can do so, one can first precompute N many triplets (h, r, s) such that $r = \overline{x_1} \bmod n$ for $(x_1, y_1) = u_1G + u_2Q$, $u_1 = \frac{h}{s} \bmod n$, and $u_2 = \frac{r}{s} \bmod n$. (This can be done by first generating u_1 and u_2, then computing r, then s and h.) Finally one hashes many messages until the hash is equal to at least one h value. (Note that we can discard all triplets for which $h \geq 2^{160}$.) On average, one needs to try $\frac{n}{N}$ messages until it works. This can therefore make another attack of **complexity** $\sqrt{n}$.

Finding collisions on SHA-1 is another threat. If we know different M_1 and M_2 such that the corresponding message digests are the same, we can ask the legitimate signer for a signature on M_1, and we get a signature for M_2. The legitimate signer can also repudiate his signature for M_1 and claim that he signed M_2... The birthday paradox provides us with an attack of **complexity** 2^{80}.

Predicting the k issued by a pseudorandom generator is another attack. If one can guess k for an (r, s) signature, then it is fairly easy to recover d. In ECDSA the pseudorandom generator uses a seed of size b bits with $160 \leq b \leq 512$. So guessing the seed by brute force attack has **a complexity of** 2^b.

Therefore, we take as necessary assumptions that

- one cannot compute discrete logarithms within less than 2^{80} computations,
- one cannot invert SHA-1 within less than 2^{160} computations,
- one cannot forge a collision without doing more than 2^{80} computations,
- one cannot predict the output from the pseudorandom generator.

We can still wonder if these conditions are sufficient for the security and if they are satisfied.

3 Dedicated Observation

Interestingly, **a valid signature can easily be transformed into another valid signature**: if (r, s) is a valid signature for M, then $(r, n - s)$ is another valid signature for M. This comes from the fact that if $(x_1, y_1) = kG$, then $(x_1, y_1') = -kG$ with $y_1' = -y_1$ if q is prime and $y_1' = y_1 + x_1$ if q is a power of two. Thus k and $-k$ generates two elliptic curve points with the same x_1 coordinate, therefore the same r, and changing k into $-k$ only changes s into $n - s$. This surprising property does not seem to endanger ECDSA.

4 SHA-1

SHA-1 is the last algorithm of series of hash functions: MD4, MD5, SHA, SHA-1. They are all based on the Merkle-Damgård paradigm [14, 23]: there is a compression function, and the message digest is computed by processing sequentially all blocks through the compression function. The Damgård result claims that if the compression function is collision resistant, then the overall construction is collision resistant as well. The compression function is made following the Davies-Meyer [15, 21] construction: each block is used as a key for an encryption function and there is a feed forward of the plaintext to the ciphertext in order to make it one-way.

MD4 [26–28] and MD5 [29] were designed by Rivest in 1990 and 1992 respectively. In 1991, Den Boer and Bosselaers [9] have found an attack against the last two round of MD4 (out of three). In 1995, Vaudenay [33] has found an attack against the first two rounds and a collision on all but a few bits. The year after, Dobbertin [16] came up with a complete collision on MD4.

MD5 has an additional round, but also a strange modification of the internal structure. This enabled Den Boer and Bosselaers [10] to forge a collision on the compression function of MD5. Although this leads to no collision on MD5 itself, it invalidates the use of the Damgård theorem.

The internal structure of SHA is closer to MD4 than to MD5, but it hashes onto 160 bits instead of 128. It also has a kind of key schedule in order to transform the processed block from one round to the other. The key schedule is however linear and parallel: the 512 bits of each block are processed as 32 parallel sequences of 16 bits. This enabled Chabaud and Joux [13] to publish an attack against SHA in 1998. The attack would enable the forgery of a collision on SHA, but requires a complexity of 2^{61} computations (which was not implemented so far).

SHA-1 is an update of SHA which was proposed before the attack of Chabaud and Joux was released without any rationale. It is believed that the NSA has found the very same attack and asked to update the standard accordingly.

Since then **no attack was found against SHA-1** and it is believed to be a secure collision resistant one-way hash function.

5 Choice of Elliptic Curves

ANSI X9.62 [2], FIPS 186 [4], and SEC2 [7] propose specific choices of elliptic curves. All choices suggested by FIPS 186 [4] are included in SEC2 [7]. Only one choice in ANSI X9.62 [2] is included in SEC2 [7] (and in FIPS 186 [4] as well): the example over a field of characteristic of 192 bits, which is called “Curve P-192” in FIPS 186 [4] and “`secp192r1`” in SEC2 [7]. ANSI X9.62 [2] proposes

- verifiable curves over the $\mathbf{F}_p$ field
- verifiable curves over the $\mathbf{F}_{2^m}$ field
- non verifiable curves based on the Weil method [2, pp. 161,165,169,171, see note 7 in Ann. A.3.2]
- curves based on unexplained method [2, p. 172]

FIPS 186 [4] proposes

- verifiable curves over the $\mathbf{F}_p$ field
- verifiable curves over the $\mathbf{F}_{2^m}$ field
- non verifiable Koblitz curves over the $\mathbf{F}_{2^m}$ field

SEC2 [7] proposes

- verifiable curves over the $\mathbf{F}_p$ field
- verifiable curves over the $\mathbf{F}_{2^m}$ field
- non verifiable Koblitz curves over the $\mathbf{F}_p$ field
- non verifiable Koblitz curves over the $\mathbf{F}_{2^m}$ field

All verifiable curves are generated by an algorithm which is standardized in ANSI X9.62 [2, Ann. A.3.3]. However, in this algorithm

- only $a^3/b^2 \bmod p$ is verifiable over $\mathbf{F}_p$ fields
- only the b parameter is verifiable over $\mathbf{F}_{2^m}$ fields.

Note that FIPS 186 [4] takes $a \equiv -3$ for $\mathbf{F}_p$-curves and $a = 1$ for $\mathbf{F}_{2^m}$-curves in all suggested parameters. SEC2 [7] takes $a \equiv -3$ for $\mathbf{F}_p$-curves in all suggested parameters but "`secp112r2`" as well, and $a = 1$ for some $\mathbf{F}_{2^m}$-curves. ANSI X9.62 [2, Ann. A.3.3] seems to pick random a in every cases.

Koblitz curves will be discussed in a separate section. Verifiable curves are discussed in the next section.

6 Generation of Public Parameters

The DSA standard suggested to generate the public parameters p, and q in a verifiable way with a pseudorandom generator, so that they are not suspected to hide a trapdoor. ECDSA specifies similar requirements for a and b, but not for q.

The elliptic curve generation for verifiable curves may look quite odd since only $a^3/b^2 \bmod p$ is verifiable in $\mathbf{F}_p$-curves and only b is verifiable in $\mathbf{F}_{2^m}$-curves. However, in the case of $\mathbf{F}_p$ fields, the $a^3/b^2 \bmod p$ characterizes the elliptic curve up to an isomorphism. More precisely, the j-invariant only depends on this quantity since

$$j = 6912 \frac{a^3/b^2}{4a^3/b^2 + 27} \bmod p.$$

(Note that $6912 = 2^8 \times 3^3$.) This means that if two curves over $\mathbf{F}_p$ have the same $a^3/b^2 \bmod p$, they must be isomorphic over the algebraic closure of $\mathbf{F}_p$. To be more precise, let us assume that we are given two curves over $\mathbf{F}_p$ defined by (a, b) and (a', b') respectively and such that

$$\frac{a^3}{b^2} \equiv \frac{(a')^3}{(b')^2} \pmod p.$$

When $p \equiv 2 \pmod 3$, b'/b can uniquely be written u^3 in $\mathbf{F}_p$. so we can write $a' = u^2 a$ and $b' = u^3 b$ for some u. Therefore the two curves are either isomorphic over $\mathbf{F}_p$ (namely when u is a quadratic residue) or twist of each other (when u is a non quadratic residue). When $p \equiv 1 \pmod 3$, this is a little more subtle since we can have $a' = u^2 va$ and $b' = u^3 b$ for some u where v is a cubic root of unity. So this can actually define six different curves depending on v and the quadratic residuosity of u. Assuming that the curve is fixed, we notice that n is unique since it is bound to the conditions of

- being prime

– being the order of at least one point G on the curve
– being greater than $4\sqrt{p}$ and than 2^{160}.

Since the order of an elliptic curve over $\mathbf{F}_p$ cannot have two different prime factors greater than $4\sqrt{p}$, n is uniquely committed to the curve. Using the structure theorem of elliptic curves, we realize that there is only one subgroup of order n. Therefore, if the curve is fixed, G can be any generator of this subgroup. Therefore, once the curve is fixed, we can only replace a, b and G by

$$\begin{aligned} a' &= u^4 a \bmod p \\ b' &= u^6 b \bmod p \\ G' &= vG \end{aligned}$$

where u and v are integers. The seed however makes two or six possible curves and n depending on $p \bmod 3$.

Similar properties hold in the case of $\mathbf{F}_{2^m}$ fields since the j-invariant depends on b only. Namely,

$$j = -\frac{1}{b}$$

in $\mathbf{F}_{2^m}$. Assuming that we are given two curves over $\mathbf{F}_{2^m}$ with the same b and defined by (a, b) and (a', b), one can first try to solve the equation $s(s+1) = a + a'$ over $\mathbf{F}_{2^m}$. It can be easily seen that this has a solution if and only if the $\mathbf{F}_{2^m}/\mathbf{F}_2$-trace of $a + a'$ is zero. In this case $x' = x$ and $y' = y + sx$ is an isomorphism between the two curves. In the other case, the two curves are actually twist of each other. Therefore the seed makes two possible curves.

In other words, **only the j-invariant of the curve is verifiable. The curve and n can be freely chosen among a choice of up to six. The particular choice of G is free.**

Notice that the public parameter modification can induce a **dramatic security problem depending on how certification is made**.

1. The certification authority signs the seed and the public key only (and a, b, G are sent separately). In this case, an attacker can just replace G by $G' = \alpha Q$ (which is not detected since G in not bound to the seed, and only the seed is certified) for a random α and sign with $d = \alpha^{-1} \bmod n$.
2. The certification authority certifies the public parameters and the public key separately. Here, there is a risk that an attacker (for instance the authority itself) can forge a certificate for related public parameters (just changing G) and attach it to the certificate for Q.
3. Discrete logarithm is easy on one of the other elliptic curve choices. (For instance, the twist of the curve is anomalous.) In this case, a honest signer can be infected by a virus who changes the a, b and G parameters in order to get an easy discrete logarithm problem. The virus can furthermore replace the certificates for the public parameters by a forged one (it is feasible to forge certificate when the discrete logarithm is easy). After the signer performs a signature, the attacker can compute the discrete logarithm of $u_1 G + u_2 Q$, get k, and extract the secret key from s.

Therefore **we recommend that the certification is made on** $(q, \mathrm{seed}, a, b, x_G, y_G, n, x_Q, y_Q)$ **with the whole public parameters and the public key all together**. Furthermore **public parameters must be kept in a secure memory which cannot be modified by viruses**.

We further raise the attention that the belief that verifiable curves are strong since they are random is not so right. Indeed, only some of the elliptic curve parameters are chosen at random, but the field is not. Therefore it is feasible to generate "at random" some weak elliptic curves by chosing p after a and b. Therefore we recommend **to update the algorithms over verifiable curves** in order to verify a, b, and G and to include q to the seed as well.

Finally, since elliptic curve validation is not so easy and since hiding trapdoors is a potential threat, we recommend that a trusted validation authority is implemented for each application: **validation of the proposed elliptic curves needs to be enforced by responsible authorities**.

7 Security for Koblitz Curves

Koblitz curves are special elliptic curves for which computations are more efficient. More precisely, Koblitz curves are made from an anomalous curve over $\mathbf{F}_q$ which is extended over $\mathbf{F}_{q^m}$. All suggested parameters from ANSI X9.62 [2] and FIPS 186 [4] take curves defined by

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

with $a = 0$ (which is anomalous over $\mathbf{F}_2$) or its twist with $a = 1$.

Note that the attack against anomalous curves does not extend (according to the present state of the art) against Koblitz curves. The benefit of this construction is that multiplication by q is much cheaper than usual (it takes a single point addition) as explained in Koblitz [20]. A drawback is that **Koblitz curves make the generic group model invalid**.

SEC1 [6, p. 62] warns that

"However despite their efficiency benefits, [Koblitz curves] should be used with a good deal of caution because they produce parameters which may be **susceptible to future special-purpose attacks**."

SEC2 [7, p. 3] also means that **no Koblitz curve are provided at export strength and at extremely high strength**.

SEC2 [7] additionally extends this notion to curves over $\mathbf{F}_p$ by

$$y^2 \equiv x^3 + b \pmod{p}$$

with $p \equiv 1 \pmod{3}$. (If we had $p \equiv 2 \pmod{3}$, this would have been an anomalous curve since the $x^3 + b$ mapping is one-to-one). However, these curves are not proposed in the ECDSA standard.

To conclude, since Koblitz curves

- are not verifiable
- invalidate the required ideal model for provable security
- are suspected to become vulnerable

we recommend not to use them.

8 Bleichenbacher's Attack

FIPS 186 [4] specifies a way to generate the one-time key k in a pseudorandom way for DSA. ANSI X9.62 [2, Ann. A.4.1] is basically the same. Namely, it constructs a pseudorandom generator who generates a 160-bit number rand and takes $k = \text{rand} \mod n$. The output rand (which is based on SHA-1 or DES) is assumed to be uniformly distributed in $[0, 2^{160} - 1]$.

Quite recently (in February 2001), Bleichenbacher announced an impressive attack based on the fact that rand $\mod n$ is not uniformly distributed in $[1, n-1]$. More precisely any k in $[2^{160} - n + 1, n - 1]$ has a probability of 2^{-160} and any k in $[0, 2^{160} - n]$ has a probability of 2.2^{-160}.

The attack was impressive enough for NIST to **release a special publication in order to recommend a new pseudorandom generator for DSA [5]**: instead of taking $k = \text{rand} \mod n$, they recommend to generate two numbers rand and rand$'$ and to take $(\text{rand}||\text{rand}') \bmod n$. Let $N = 2^{320}$. Here any k in $[(N \bmod n) + 1, n - 1]$ has a probability of $\lfloor \frac{N}{n} \rfloor N^{-1}$ and any k in $[0, N \bmod n]$ has a probability of $(\lfloor \frac{N}{n} \rfloor + 1)N^{-1}$. Both probabilities are between $n^{-1} - N^{-1}$ and $n^{-1} + N^{-1}$.

Bleichenbacher's attack uses a bias of k defined by

$$\text{bias}(k) = E\left(e^{\frac{2i\pi k}{n}}\right).$$

When $k = \text{rand} \mod n$ with rand uniformly distributed in $[0, N-1]$, we have

$$\begin{aligned}
\text{bias}(k) &= \sum_{r=0}^{N-1} \frac{1}{n} e^{\frac{2i\pi(r \bmod n)}{n}} \\
&= \sum_{r=0}^{N-1} \frac{1}{n} e^{\frac{2i\pi r}{n}} \\
&= \frac{1}{N} \times \frac{e^{\frac{2i\pi N}{n}} - 1}{e^{\frac{2i\pi}{n}} - 1} \\
&= \frac{e^{i\pi\frac{N-1}{n}}}{N} \times \frac{\sin(\frac{\pi N}{n})}{\sin(\frac{\pi}{n})} \\
&\approx \frac{ne^{i\pi\frac{N-1}{n}}}{\pi N} \times \sin(\frac{\pi N}{n})
\end{aligned}$$

Obviously, when N is close to n, say $n = N - \varepsilon$, we have $\text{bias}(k) \approx -\frac{\varepsilon}{N} e^{i\pi\frac{N-1}{n}}$ which is small. For primes n arbitrarily chosen of size 160 and $N = 2^{160}$, the sinus can be quite large and the norm of the bias can be within the order of magnitude of $\frac{1}{\pi}$. When N is within the order of magnitude of n^2 (as in the fix recommended by NIST), the norm is less than n^{-1}.

In the case of ECDSA, n is close to 2^{160} in most of proposed elliptic curves, so the attack is not directly applicable. The fix proposed by NIST for DSA is not yet proposed for ECDSA. However, some elliptic curve choices may still lead to a similar attack, so we recommend **to update the pseudorandom generator**, at least in a similar manner than was was done in [5].

Our comment is that both Bleichenbacher's attack and NIST's recommendation are still underground research: none have been published in the academic world yet, and the impact and limitations of the cryptanalytic branch opened by Bleichenbacher are not yet quite clear. Although the NIST fix looks quite reasonable, wisdom recommends to **let the pseudorandom generator open** in the standard for later release in case of new research results.

9 Pseudorandom Generator for Secret Keys and k Values

FIPS 186 [4] suggests that random numbers must be generated according to a deterministic pseudorandom generator. These values include the secret keys d and the one-time keys k. FIPS 186 [4] and the fix to Bleichenbacher attack [5] can be described as follows.

1. We construct a primitive G who takes two inputs: a 160-bit parameter t (the *type*) which is a constant specific to the number we want to generate (d or k), and a b-bit ($160 \leq b \leq 512$) key which is also specific to the number we want to generate. The primitive outputs a 160-bit number. Two constructions for G are proposed.
   The first one is based on the compression function of SHA-1. This means that we do not consider message padding, iteration, and initialization as in the Merkle-Damgård [14, 23] construction. The registers of the compression function are initialized with the type value. The message block consists of the key padded with zero bits up to 512 bits. The output of the compression function (obtained after the feed forward due to the Davies-Meyer scheme [15, 21]) is the output of G.
   The second one is based on DES. It consists of the following steps.
$$\begin{aligned} t &= t_1||t_2||t_3||t_4||t_5 \\ c &= c_1||c_2||c_3||c_4||c_5 \\ x_i &= t_i \oplus c_i \\ y_{i,1}||y_{1,2} &= \mathrm{DES}_{c_{i+4}||c_{i+3}}(x_i||(x_{i+1} \oplus x_{i+4})) \\ z_i &= y_{i,1} \oplus y_{i+2,2} \oplus y_{i+3,1} \\ G(t,c) &= z_1||z_2||z_3||z_4||z_5 \end{aligned}$$
   with $i = 1, 2, 3, 4, 5$ where subscripts are considered modulo 5.
2. We construct from G a pseudorandom generator which generates 160-bit numbers which is specific to a given type t and we denote rand_t. The generator can use a user defined input which is called seed, and uses a seed which is surprisingly called key_t. It works as follows.
   (a) $\mathrm{rand}_t \leftarrow G(t, \mathrm{key}_t + \mathrm{seed} \bmod 2^b)$
   (b) $\mathrm{key}_t \leftarrow 1 + \mathrm{key}_t + \mathrm{rand}_t \bmod 2^b$
3. We construct a specific pseudorandom generator. For FIPS 186 [4], it simply consists in taking $\mathrm{rand}_t \bmod q$. For [5], it consists in calling the pseudorandom twice in order to get rand_t and rand'_t and taking $(\mathrm{rand}_t||\mathrm{rand}'_t) \bmod q$.

Note that **the system is vulnerable against many kinds of replay attacks**. If the signer is a tamper proof device, assuming that we can clone it, then we can ask the two clones to sign different messages. We will then get two s signatures with the same unknown k and d and for different messages. It is then easy to recover d. If the signer's seed is in a hard disk (encrypted or not), it may be feasible to restore the hard disk in one of its previous state (either by the attack or by a restore from backup due to computer crash). We can also get two signatures which use the same k values this way. **Implementation therefore needs extra care.**

Considering the pseudorandom generator as a random function with an internal state of b bits, we know that the expected loop length is within the order of magnitude of $\sqrt{2^b}$. For $b = 160$, we start having **security problems after about 2^{80} signatures**.

For the specific pseudorandom generators proposed in FIPS 186 [4], we observed that **the DES-based one has strange properties**. For instance, if all t_i are equal, as well as all c_i, then the generated $G(t, c)$ has the same property. More interestingly, the cyclic shift by 32 bits R has the property that $G(R(t), R(c)) = R(G(t, c))$. In particular, since the t value of the generator for k is obtained by R on the t value for the generator for d, if both seeds are related by R, the two generators generate x and k such that $k = R(d)$... Although this does not seem to be dangerous for ECDSA, it is obviously the kind of remarkable property that a pseudorandom generator should not have. Therefore we think that **the DES-based pseudorandom generator is not mature enough** in the academic sense.

10 Security of ECDSA in an Ideal Model

Two different security results in ideal models are available from Pointcheval and Vaudenay [25], and Brickell et al. [11]. In both, we consider SHA-1 as a random oracle H. In the first one, we also consider $r = H_1(kG)$ with a random oracle H_1 instead of $r = \overline{x_1} \bmod n$ with $(x_1, y_1) = kG$. Hypothesis are

- *the random oracles H and H_1 are uniformly distributed*
- *the output of H and H_1 are subsets of* $[0, n - 1]$.

The first step of the security result is as follows.

> *An adaptive attack which can output an existential signature forgery can be transformed into an adaptive attack which extract the secret key d.*

The result is also quantitative.[1] Note that the result is quite idealistic. Furthermore, this step "only" proves that signature forgery is equivalent to secret key awareness. But there in no clue about information leakage from the signature. Namely, it does not prove that finding the secret key with an adaptive attack is hard. This is actually not the case without any additional assumption on the randomness of k as demonstrated by Bleichenbacher.

Nevertheless, a second step can be performed with the additional assumption

- *all k values are independently and uniformly generated in $\mathbf{Z}_n^*$ (or indistinguishable from such a generator).*

With this we prove that we can simulate the signature oracle, which proves the following result.

> *An adaptive attack which can output an existential signature forgery can be transformed into an algorithm which computes the secret key d.*

[1] See [25] or [11] for more information.

Here, access to the signing oracle is removed. Therefore, **the signature is provably secure provided that H and H_1 can be approximated by random oracles**, that the generator for k is secure, and that the discrete logarithm problem is hard.

In the second result, we assume that the modulo H_1 mapping from the group spanned by G to $[0, q-1]$ has no ℓ-collision (namely ℓ pairwise different inputs which produce the same output) for $\ell = O(\log n)$, and that $H(M)$ is replaced by $H(r, M)$ as in the Schnorr signature [30, 31]. (So this is not the standard scheme.) Hypothesis and security results are similar. Although the hypothesis on H_1 is reasonable, the signature scheme has been modified so the result is not applicable for ECDSA.

Brown [12] also proved the security in another model. In his model, **group operations are performed by a random oracle**, but H_1 and SHA-1 are fixed. He also assumes that H_1 is invertible (which is the case for ECDSA). Brown basically proves that **signature forgery is equivalent to inverting SHA-1 for passive attacks** (i.e. no-message attacks) and **to finding collisions for active attacks**. (Provided that pseudorandom generators are good!) The generic group model is quite realistic for ECDSA **assuming that we have "random" elliptic curves** since we basically know no better algorithms than generic ones for discrete logarithm computation. (See Shoup [32] and Maurer-Wolf [22].)

Our comment on this model is that generic group assumption simply means that we do not master elliptic curve tricks. However it is reasonable to believe that within a few years, education and research will bring a sufficient understanding on elliptic curve computations and that algorithms better than generic ones will be discovered. In short, **we believe that the generic group assumption is a good model for our lack of understanding of elliptic curves nowadays**. It is therefore a reasonable model at this time, but **there is no clue how long it will remain so**.

Particular elliptic curves are known to be subject to specific attacks. Namely supersingular and anomalous elliptic curves make the discrete logarithm problem easy. More generally, if n divides $q^i - 1$ with i small, the MOV algorithm enables the computation of discrete logarithm (see [8]). This is usually avoided by checking that the order of q in $\mathbf{Z}_n^*$ is large (namely greater than 20). It should be outlined that these specific attacks are very recent research achievements and that **other specific attacks are likely to be discovered in the future**. Therefore selecting elliptic curves for which the discrete logarithm problem is hard is a tough question. The SEC2 [7, p. 4] standard suspects that **hiding trap doors in elliptic curve could be feasible**. Therefore it recommends to construct elliptic curve in a deterministic pseudorandom way and to publish the algorithm and the chosen seed.

11 Conclusion

We have shown that ECDSA is vulnerable against attacks of complexity $\sqrt{n}$. The pseudorandom generator for keys and one-time keys has just been changed for security reasons for DSA. It would be nice to do the same for ECDSA. Since this has not been addressed by the academic community at the time this report was written, it would be wise to prepare potential modifications of this part in the future.

We pointed out an unexpected property of the DES-based G generator. Therefore we think that this part of the standard must be reconsidered. In the meantime we recommend to use the SHA-based generator.

We also outline that implementations may easily lead to security issues by kinds of replay attacks due to the deterministic pseudorandom generator. This would deserve informative security warnings in the standard.

ECDSA is provably secure in the random oracle model, provided

1. that the generator for k is secure,
2. that the SHA-1 hash function can be approximated by a random oracle,
3. that the discrete logarithm is hard in the corresponding group,
4. and that we approximate the mapping from the exponential of k to r by a random oracle.

The first assumption was wrong as shown by Bleichenbacher. The third assumption is questionable since very few people in the world really understand the algorithmic properties of elliptic curves. The author of the present report does not claim to be in this category but believes that future research may raise surprising results. The second and last assumptions are however quite artificial.

ECDSA is provably secure in the generic group model, provided

1. that the generator for k is secure,
2. and that the SHA-1 hash function is collision-resistant.

(We know that the discrete logarithm is hard in this model, so there is no reason to assume it further.) It is acceptable at this time provided that the selected elliptic curve is a random one (which excludes Koblitz curves) since the understanding of elliptic curve arithmetic is not so popular.

Due to the doubts raised by elliptic curve experts on the security of Koblitz elliptic curves in the long term, we recommend not to use them.

ECDSA faces to the problem of parameters validation: parameters need to be authentic and trapdoor-free. For this a pseudorandom generation algorithm needs to be specified. This was half done since q, G, and specific choice of a and b are missing. Hence we recommend to update the public parameter selection algorithms: ECDSA should generate all parameters: q, a, b, G, ... We further recommend to require elliptic curve validation by the authority (and not only to take samples from the standards).

References

1. ANSI X9.30. Public Key Cryptography for the Financial Services Industry: Part 1: The Digital Signature Algorithm (DSA). American National Standard Institute. American Bankers Association. 1997.
2. ANSI X9.62. Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard Institute. American Bankers Association. 1998.
3. Secure Hash Standard. *Federal Information Processing Standard* publication #180-1. U.S. Department of Commerce, National Institute of Standards and Technology, 1995.
4. Digital Signature Standard (DSS). *Federal Information Processing Standards* publication #186-2. U.S. Department of Commerce, National Institute of Standards and Technology, 2000.
5. Recommendations Regarding Federal Information Processing Standard (FIPS) 186–2, Digital Signature Standard (DSS). NIST Special Publication 800–XX. U.S. Department of Commerce, National Institute of Standards and Technology, October 2001.

6. SEC 1: Elliptic Curve Cryptography. v1.0, Certicom Research, 2000.
`http://www.secg.org/secg_docs.htm`
7. SEC 2: Recommended Elliptic Curve Cryptography Domain Parameters. v1.0, Certicom Research, 2000.
`http://www.secg.org/secg_docs.htm`
8. R. Balasubramanian, N. Koblitz. The Improbability that an Elliptic Curve has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *Journal of Cryptology*, vol. 11, pp. 141–145, 1998.
9. B. den Boer, A. Bosselaers. An Attack on the Last two Rounds of MD4. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 576, pp. 194–203, Springer-Verlag, 1992.
10. B. den Boer, A. Bosselaers. Collisions for the Compression Function of MD5. In *Advances in Cryptology EUROCRYPT'93*, Lofthus, Norway, Lectures Notes in Computer Science 765, pp. 293–304, Springer-Verlag, 1994.
11. E. Brickell, D. Pointcheval, S. Vaudenay, M. Yung. Design Validations for Discrete Logarithm Based Signature Schemes. In *Public Key Cryptography*, Melbourne, Australia, Lectures Notes in Computer Science 1751, pp. 276–292, Springer-Verlag, 2000.
12. D.R.L. Brown. The Exact Security of ECDSA. Technical Report CORR 2000–34, Certicom Research, 2000.
`http://www.cacr.math.uwaterloo.ca`
13. F. Chabaud, A. Joux. Differential Collisions in SHA-0. In *Advances in Cryptology CRYPTO'98*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 1462, pp. 56–71, Springer-Verlag, 1998.
14. I. B. Damgård. A Design Principle for Hash Functions. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.
15. R. W. Davies, W. L. Price. Digital Signature – an Update. In *Proceedings of the International Conference on Computer Communications*,Sydney, pp. 843-847, North-Holland, 1985.
16. H. Dobbertin. Cryptanalysis of MD4. In *Fast Software Encryption*, Cambridge, United Kingdom, Lectures Notes in Computer Science 1039, pp. 53–69, Springer-Verlag, 1996.
17. T. ElGamal. Cryptography and Logarithms over Finite Fields. PhD Thesis, Stanford University, 1984.
18. T. ElGamal. A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms. In *Advances in Cryptology CRYPTO'84*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 196, pp. 10–18, Springer-Verlag, 1985.
19. T. ElGamal. A Public-key Cryptosystem and a Signature Scheme based on Discrete Logarithms. *IEEE Transactions on Information Theory*, vol. IT-31, pp. 469–472, 1985.
20. N. Koblitz. CM-Curves with good Cryptographic Properties. In *Advances in Cryptology CRYPTO'91*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 576, pp. 279–287, Springer-Verlag, 1992.
21. S. M. Matyas, C. H. Meyer, J. Oseas. Generating Strong One-way Functions with Cryptographic Algorithm. *IBM Technical Disclosure Bulletin*, vol. 27, pp. 5658–5659, 1985.
22. U. Maurer, S. Wolf. Lower Bounds on Generic Algorithms in Groups. In *Advances in Cryptology EUROCRYPT'98*, Espoo, Finland, Lectures Notes in Computer Science 1403, pp. 72–84, Springer-Verlag, 1998.
23. R. C. Merkle. One way Hash Functions and DES. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 416–427, Springer-Verlag, 1990.
24. V.I. Nechaev. Complexity of a Determinate Algorithm for the Discrete Logarithm. Mathematical Notes of the Academy of Sciences of the USSR, vol. 55 pp. 165–172, 1994.
25. D. Pointcheval, S. Vaudenay. On Provable Security for Digital Signature Algorithms. Technical report LIENS 96-17, Ecole Normale Supérieure, 1996.
26. R. L. Rivest. The MD4 Message Digest Algorithm. In *Advances in Cryptology CRYPTO'90*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 537, pp. 303–311, Springer-Verlag, 1991.
27. R. L. Rivest. The MD4 Message Digest Algorithm. RFC 1186, Oct 1990.
28. R. L. Rivest. The MD4 Message Digest Algorithm. RFC 1320, Apr 1992.
29. R. L. Rivest. The MD5 Message Digest Algorithm. RFC 1321, Apr 1992.
30. C. P. Schnorr. Efficient Identification and Signature for Smart Cards. In *Advances in Cryptology CRYPTO'89*, Santa Barbara, California, U.S.A., Lectures Notes in Computer Science 435, pp. 235–251, Springer-Verlag, 1990.
31. C. P. Schnorr. Efficient Identification and Signature for Smart Cards. *Journal of Cryptology*, vol. 4, pp. 161–174, 1991.
32. V. Shoup. Lower Bounds for Discrete Logarithms and Related Problems. In *Advances in Cryptology EUROCRYPT'97*, Konstanz, Germany, Lectures Notes in Computer Science 1233, pp. 256–266, Springer-Verlag, 1997.
33. S. Vaudenay. On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In *Fast Software Encryption*, Leuven, Belgium, Lectures Notes in Computer Science 1008, pp. 286–297, Springer-Verlag, 1995.