Security Level of Cryptography —

ECDSA

December 2001

Contents

Summary

This report analyzes the security of the ECDSA signature scheme [1, 10, 14]. We analyze the security of ECDSA against existential forgery under a chosen-message attack. The analysis is done in the generic-group model and is primarily based on a result of Brown [5]. We explain the implications of this model and relate the security analysis to the real-world security of ECDSA. We also consider the security of ECDSA from an implementation viewpoint. We discuss the importance of using a proper pseudorandom generator (PRG) and review some attacks on ECDSA when the PRG is used improperly. The second part of the report analyzes the security of discrete-log on the group of points of an elliptic curve. We focus on Koblitz curves and discuss some recent attacks on these curves.

Our conclusions are as follows:

- The analysis in Brown's paper [5] is incomplete. The proof of the main theorem (Theorem 2) showing that ECDSA is secure against existential forgery under a chosen-message attack is very sketchy. A few other omissions and inaccuracies are given in Section 9 of this report.
- We give precise definition for the generic-group model and its applicability to ECDSA. We also state without proof the main theorem from [5] showing that ECDSA in the generic-group model is secure against existential forgery under a chosen-message attack assuming SHA-1 is collision resistant and uniform.
- Plugging-in actual values into the main theorem shows that in the generic-group model, ECDSA has adequate security.
- We discuss the implications of the generic-group model to the real world and emphasize that security in the generic-group model does not imply security in the real world.
- We discuss attacks on the implementation of ECDSA. We emphasize that ECDSA requires a strong pseudorandom generator, capable of generating independent and uniformly distributed elements in the range $[0, n-1]$ for some n.
- Finally, we conclude with a discussion of Koblitz curves. We survey some recent attacks on Koblitz curves, but conclude that these attacks do not affect the security of these curves. Our current knowledge indicates that Koblitz curves offer adequate security and can be deployed in real systems.

This report was prepared at the request of the Information-Technology Promotion Agency, Japan.

1 Signature Schemes

We begin by explaining what constitutes a signature scheme and what it means for one to be secure. A (fixed security parameter) *digital signature scheme* is a tuple of three randomized algorithms $\Pi = \langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$. These algorithms work as follows:

Gen: Takes no input and outputs a public-key PK and a private key SK.

Sign: Takes a message $M \in \{0,1\}^*$ and a private key SK and generates a signature S. We write $\mathsf{Sign}(M, \mathsf{SK}) = S$.

Verify: Takes a message $M \in \{0,1\}^*$, a signature S, and a public key PK. The algorithm outputs true or false. We write $\mathsf{Verify}(M, S, \mathsf{PK}) \in \{\mathsf{true}, \mathsf{false}\}$.

The only consistency requirement is that $\mathsf{Verify}(M, S, \mathsf{PK}) = \mathsf{true}$ whenever $S = \mathsf{Sign}(M, \mathsf{SK})$. Here $\langle \mathsf{PK}, \mathsf{SK} \rangle$ is some public/private key pair output by Gen.

Next, we define what it means for a signature scheme to be secure [8]. To do so we define a game involving a signature-forging algorithm FG and a challenger algorithm. The game runs in three phases:

Init: The challenger runs the Gen algorithm to get a pair $\langle \mathsf{PK}, \mathsf{SK} \rangle$. It sends PK to the forger FG and keeps SK to itself.

Queries: The forger FG mounts an adaptive chosen-message attack by querying the challenger with messages $M_1, \ldots, M_q \in \{0,1\}^*$ of its choice. The challenger responds with signatures $S_i = \mathsf{Sign}(M_i, \mathsf{SK})$ for $i = 1, \ldots, q$. Note that FG may issue the queries adaptively, i.e. query M_{i+1} may depend on the messages and signatures $M_1, S_1, \ldots, M_i, S_i$.

Output: Eventually the forger FG outputs a message/signature pair $\langle M, S \rangle$. We say that FG won the forging game if $\mathsf{Verify}(M, S, \mathsf{PK}) = \mathsf{true}$ and $M \neq M_i$ for all $i = 1, \ldots, q$. In other words, FG forged a signature on some new message M. We refer to $\langle M, S \rangle$ as a forgery.

We say that FG is a (t, ϵ, q)-forger if its total running time is t, it issues at most q chosen-message queries, and it wins the game with probability at least ϵ. The probability space is defined over the random bits used by the challenger and the forger FG. We say that a signature scheme is (t, ϵ, q) secure against existential forgery under an adaptive chosen-message attack if there is no (t, ϵ, q)-forger. We often use shorthand and simply say that the signature scheme is (t, ϵ, q) secure.

A signature scheme which is $(t, \epsilon, 0)$ secure is said to be secure against existential forgery under a passive attack. The word "passive" refers to the fact that the scheme might only be secure against forgers FG that make no chosen-message queries.

2 The ECDSA Signature Scheme

We describe a generalized ECDSA signature scheme called GenericDSA and then we explain how to derive ECDSA from this scheme [5]. The scheme uses the following fixed public parameters:

1. A group $\mathbb{G}$ of prime order n, and a generator $P \in \mathbb{G}$. We view $\mathbb{G}$ as an additive group.
2. An efficiently computable hash function $H : \{0,1\}^* \to [0, n-1]$.

3. An efficiently computable reduction function $F : \mathbb{G} \to [0, n-1]$.

We describe the signature scheme GenericDSA by describing the three algorithms $\langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$:

Gen: Pick a random integers $d \in [0, n-1]$. Set $Q = dP \in \mathbb{G}$.
The public key is Q. The private key is d.

Sign: To sign a message $M \in \{0,1\}^*$ using the private key d, do the following:

1. Pick a random integer $k \in [1, n-1]$.
2. Compute $R = kP \in \mathbb{G}$.
3. Compute $r = F(R)$ and $e = H(M)$. Note $r, e \in [0, n-1]$.
4. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to step 1.
5. Output $S = \langle r, s \rangle$ as the signature on M.

Verify: To verify a signature $S = \langle r, s \rangle$ on a message M using the public key Q check that:

$$r \stackrel{?}{=} F\left(s^{-1}H(M) \cdot P + s^{-1}r \cdot Q\right)$$

ECDSA is an instantiation of GenericDSA obtained by taking
(1) $\mathbb{G}$ as the group of points of order n on an elliptic curve $E/\mathbb{F}_q$ approved by NIST [14];
(2) taking H as the hash function SHA-1 whose output is reduced modulo n; and
(3) taking $F : E/\mathbb{F}_p \to [0, n-1]$ as the function defined by $F(R) = R_x \bmod n$ where R_x is the x-coordinate of the point $R \in E/\mathbb{F}_p$ viewed as an integer in $[0, p-1]$. Define $F(O) = 0$ where O is the point at infinity on $E/\mathbb{F}_p$.

3 The Generic Model

Brown [5] studies the security of GenericDSA (and hence of ECDSA) in the generic-group model. In this section we precisely define the generic-group model as needed for Brown's analysis. Later we will discuss the significance of this model to the security of ECDSA.

The generic-group model, due to Nechaev [13] and Shoup [15], is motivated by the observation that known attacks on elliptic-curve cryptosystems (using one of the approved NIST curves [14]) do not make use of the representation of points on the curve. Instead, the attack algorithms only perform group operations on points on the curve. Consequently, known attacks work no matter what underlying group is used to define the cryptosystem. An attack algorithm FG of this type can be abstracted so that it only uses group addition and subtraction as a black box—the algorithm FG submits strings representing two group elements and gets back a string representing their sum or difference.

More precisely, the generic-group model used in Brown's security proof is defined as follows:

1. The model is parameterized by a prime number n and a set of strings $S \subseteq \{0,1\}^*$ of cardinality n. The strings in S will represent elements of some group $\mathbb{G}$ which is isomorphic to $\mathbb{Z}_n$. Both n and S are fixed and are given to the attack algorithm FG. (The sense in which the set S is given to FG is via an oracle that can be used to test membership in S, as discussed below.)
2. Let σ be a one-to-one map $\sigma : \mathbb{Z}_n \to S$. Then σ induces a group action on S by defining

$$\forall X, Y \in S : \quad X + Y = \sigma(\sigma^{-1}(X) + \sigma^{-1}(Y))$$

   The identity element of S is the string $\mathbf{0} = \sigma(0)$ and the inverse of $X \in S$ is $\sigma(-\sigma^{-1}(X))$.
3. An algorithm $\mathcal{A}$ manipulates group elements using an oracle $\mathcal{O}_\sigma$ with the following behavior:
   - **Subtraction:** Given $X, Y \in S$ the oracle returns $X - Y \in S$ where subtraction is defined via the map σ as described above. If one of X, Y is not in S the oracle returns $\mathsf{invalid}$.
   - **Sampling:** Given a query of ε (the empty string), the oracle returns a random string X from S.

Note that the ability to perform subtraction is sufficient for obtaining the identity $\mathbf{0} = \sigma(0) = X - X$, for adding two elements $X + Y = X - (\mathbf{0} - Y)$, and for inverting an element $-X = \mathbf{0} - X$. Using addition and repeated doubling one can also compute $\ell \cdot X$ for any $X \in S$ and $\ell \in \mathbb{Z}$. Finally, according to our conventions, the oracle can also be used to test membership in S: to test if $X \in S$ ask the oracle to compute $\mathbf{0} - X$ and see whether or not it returns $\mathsf{invalid}$.

Different maps $\sigma : \mathbb{Z}_n \to S$ induce different group actions on S. Let S_σ be the group induced by σ. The point is that if $\mathcal{A}$ performs a successful attack that treats the underlying group as a black-box then $\mathcal{A}$ should perform well if σ is chosen uniformly at random from the finite set of all maps $\sigma : \mathbb{Z}_n \to S$.

When we speak of working in the generic-group model we have embellished the model of computation so as to give the adversary $\mathcal{A}$ an oracle for computing the group action in S_σ, for some σ. We also allow S_σ to influence the behavior of the schemes that we are considering, as described in the next section for the case of a digital-signature scheme in the generic-group model.

4 Generic-Model Signature Schemes and their Security

We now define what it means for a signature scheme to be secure in the generic-group model. First, a signature scheme in the generic-group model is a standard signature scheme $\langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$ except that these three algorithms are allowed to use the generic-group oracle above (for sampling or subtraction). For example, to turn GenericDSA into a signature scheme in the generic-group model we simply set $\mathbb{G} = S_\sigma$ for some set S of cardinality n. Note that the algorithms $\langle \mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify} \rangle$ of GenericDSA can be implemented using only the generic-group oracle $\mathcal{O}_\sigma$. The reduction function $F : \mathbb{G} \to [0, n-1]$ is defined as some function $F : S \to [0, n-1]$ and is independent of σ.

Security is defined using almost the same game as in the previous section. The only difference is that the group oracles are defined using a map $\sigma : \mathbb{Z}_n \to S$ chosen uniformly at random from (the finite set of) one-to-one maps $\mathbb{Z}_n \to S$, and both the challenger and forger FG are allowed to interact with the generic-group oracle $\mathcal{O}_\sigma$.

We say that FG is a (t, ϵ, q, q_G)-forger if its total running time is t, it issues at most q chosen-message queries and q_G queries to the group oracle $\mathcal{O}_\sigma$, and it wins the game with probability at least ϵ. Here the probability space is defined over the random choice of $\sigma : \mathbb{Z}_n \to S$ and the random bits used by the challenger and the forger FG.

We say that a signature scheme is (t, ϵ, q, q_G) secure against existential forgery under an adaptive chosen-message attack in the generic-group model if there is no (t, ϵ, q, q_G)-forger. We often use shorthand and simply say that the signature scheme is (t, ϵ, q, q_G) secure in the generic-group model.

Comparison with Shoup's generic model. We note that the generic-group model defined above differs from Shoup's model [15] in that it enables an algorithm $\mathcal{A}$ to sample random elements from S. Shoup's model does not allow that. Allowing $\mathcal{A}$ to sample random elements models elliptic curve groups more accurately since in such groups sampling a random group element can be efficiently found.

Another difference is that we fix the set S once and for all as opposed to letting S be random. In Shoup's model σ is a random injective map $\sigma : \mathbb{Z}_n \to \{0,1\}^k$ for a sufficiently large k. This means that group elements are represented using random k-bit strings. In the model above group elements are represented using random elements from an n-element set S. This restriction seems necessary for Brown's proof of security since otherwise it is not clear how to invert the reduction function $F : S \to [0, n-1]$. On the positive side the restriction enables us to make concrete statements about the real ECDSA scheme — we can argue about security of GenericDSA using the reduction map F of the real ECDSA. To do so we make S the set of representations of points of an elliptic curve $E/\mathbb{F}_p$. Then the reduction map $F : E/\mathbb{F}_p \to [0, n-1]$ of ECDSA is a valid reduction map and is well defined independently of σ. The generic-group model analysis now applies to ECDSA with its actual F and H functions. The only thing being abstracted by the generic model is the group oracle.

5 The Significance of Generic-Model Results

Analyzing practical signature scheme using concrete complexity assumptions has turned out to be quite hard. In fact, none of the standardized schemes [14] (e.g., RSA, DSA, ECDSA) has been shown to be existentially unforgeable under a chosen-message attack based on standard complexity assumptions such as the difficulty of factoring or discrete-log. Nevertheless, obtaining good assurance requires some sort of security argument for these standardized and deployed schemes.

A common approach, initiated by Bellare and Rogaway [2], for analyzing practical systems is to prove security against a restricted class of forgers. The hope is that this restricted class is generic enough to capture all common attack techniques. Bellare and Rogaway proposed to prove security against forgers restricted to the random oracle model. Roughly speaking, a signature scheme secure in the random-oracle model is secure against all forgers that treat the hash function used in the scheme as a black-box (the random-oracle model makes this statement precise). The beauty of the random-oracle model is that it captures virtually all known methods of attacking practical signature schemes. In other words, if a signature scheme is secure in the random-oracle model then it essentially resists all common attack methods.

Unfortunately, the statements above over simplify the state of things. Suppose a signature is secure against all forgers that treat the hash function as a black-box (more precisely, the signature scheme is secure in the random-oracle model). It is still possible that there is a successful forger that

does not treat the hash function as a black-box. In fact, it is possible that for every instantiation of the black-box there is a successful forger. As discussed above, there are no known attacks on practical signatures schemes that fall into this category. Nevertheless, it is possible to construct an artificial signature scheme which is secure in the random-oracle model and yet for every instantiation of the hash function there is a successful forger. Canetti, Goldreich, and Halevi [6] describe such a signature scheme. This result helps emphasize that proofs of security in the random-oracle model should be interpreted with care: although the system will resist all forgers that treat the hash function as a black-box, it might not resist more general forgers.

The random-oracle model has been very successful in providing security arguments for practical cryptosystems. Often there is no other way to argue about the security of these systems. Clearly, proving security against the most common class of forgers is far better than no security argument at all. As a result, the random-oracle model has become a valuable tool in the design of practical cryptosystems.

Unfortunately, nobody has, so far, been able to prove security of the ECDSA in the random-oracle model. Hence the effort to prove security against another class of restricted forgers. Roughly speaking, a signature scheme secure in the generic-group model is secure against all forgers that treat the underlying group as a black-box (we formalized this notion in the previous section).

The generic-group model seems less natural than the random-oracle model. Cryptographic hash functions such as SHA-1 are designed to mimic a random function $h : \{0,1\}^* \to \{0,1\}^{160}$. Hence, replacing the random-oracle by SHA-1 is quite natural. On the other hand, group elements are not designed to look like random strings. Hence, it is not obvious why replacing the generic-group with a concrete group will retain the security properties. To give a concrete example, consider a signature scheme secure in the generic-group model. When the generic group is instantiated using the group $\mathbb{F}_p^*$ the resulting system is likely to be completely insecure. The generic-group argument might show that the system is sufficiently secure when the group size is $n = 2^{160}$. However, using $\mathbb{F}_p^*$ with a prime of order 2^{160} is likely to result in an insecure system.

Nevertheless, if we instantiate the signature scheme with certain other groups the resulting system does seem to have the predicted security. Commonly two classes of groups are used to instantiate constructions in the generic-group model:

1. Small subgroups of $\mathbb{F}_p^*$ of order n. The prime p should be much larger than n, e.g. $p > n^2$.
2. Subgroups of order n of an elliptic curves over a finite field $\mathbb{F}_q$. Not all elliptic curves can be used for proper security. Therefore, one should only instantiate generic-group constructions using carefully considered curves, like those approved by NIST [14].

We can heuristically claim that instantiating a generic-group construction with one of these two groups retains the security properties of the system. If no other security analysis is possible, a proof in this model may be the best that one can do, and certainly does do something to engender confidence.

We note that the Canetti, Goldreich, and Helevi [6] example discussed above can be adapted to the generic-group model as well. One can build a signature scheme secure in the generic-group model but insecure when the group is instantiated with a specific elliptic curve. Consequently, as before, one must be careful not to read too much (or too little) into generic-model conclusions.

6 Security of ECDSA in the Generic Model

We review the main security result of Brown [5] regarding the security of ECDSA in the generic-group model. We make the result more precise and correct some mistakes in the error estimates (see Section 9). We begin with an analysis of GenericDSA and the apply the results to ECDSA. Before stating the main results we need to define some auxiliary concepts.

An α-uniform distribution. Let $\mathcal{P}$ be a distribution on some set S. We say that $\mathcal{P}$ is *α-uniform on S* if

$$\forall A \subseteq S: \quad \frac{1}{\alpha} \leq \frac{\Pr_{\mathcal{P}}[A]}{|A|/|S|} \leq \alpha$$

In other words, the ratio of the weight of A under $\mathcal{P}$ and the weight of A under the uniform distribution is bounded above and below by α and $1/\alpha$, respectively.

The notion of (t, α)-almost-invertibility. We say that the reduction function $F : \mathbb{G} \to [0, n-1]$ is (t, α)-*almost-invertible* if there exists a t-time randomized algorithm called F^{-1} such that:

1. F^{-1} takes as input $b \in [0, n-1]$ and either outputs an element of $\mathbb{G}$ or outputs invalid.
2. When $b \in [0, n-1]$ is uniformly random we have that $\Pr_b[F(F^{-1}(b)) = b] > 1/3$.
3. When $b \in [0, n-1]$ is uniformly random the distribution induced on $\mathbb{G}$ by $F^{-1}(b)$ is α-uniform on $\mathbb{G}$.

For example, let $\mathbb{G}$ be the group of points of size $n \approx 2^{160}$ on an elliptic curve $E/\mathbb{F}_p$ where $p \approx 2^{160}$. Recall that the ECDSA reduction function $F : \mathbb{G} \to [0, n-1]$ maps a point $R = (x, y) \in \mathbb{G}$ to $F(R) = x \bmod n$ (and $F(O) = 0$). We claim that this map is $(5, (\log n)^3)$-almost-invertible. To see this define the algorithm F^{-1} on input $x \in [0, n-1]$ as follows:

(1) F^{-1} builds a list L of all points $R = (x', y') \in \mathbb{G}$ such that $x' = x \bmod n$ (there are at most 4 such points). If $x = 0$ add the point at infinity O to the list.

(2) F^{-1} outputs a random point $R \in E/\mathbb{F}_p$ from the list L. If the list L is empty F^{-1} outputs invalid.

When $F^{-1}(x) \in \mathbb{G}$ we clearly have $F(F^{-1}(x)) = x$. Furthermore, since the inverse image of each $x \in [0, n-1]$ under F has size at most 5 it follows that F outputs a 5-uniform distribution on $E/\mathbb{F}_p$. The running time is $(\log n)^3$ since F^{-1} must compute square roots in $\mathbb{F}_p$. Hence, F is $(5, (\log n)^3)$-almost-invertible.

Collision finder. Let $H : \{0,1\}^* \to [0, n-1]$ be a hash function. A (t, ϵ) *collision finder for H* is a randomized algorithm that runs in time t and with probability at least ϵ outputs a pair $\langle M_0, M_1 \rangle$ with $M_0 \neq M_1$ such that $H(M_0) = H(M_1)$. (The notion is meaningful despite the fact that an $(O(\lg n), 1)$-collision-finder exists for any $H : \{0,1\}^* \to [0, n-1]$.)

The notion of (t, γ)-uniform hashing. A hash function $H : \{0,1\}^* \to [0, n-1]$ is said to be (t, γ)-*uniform* if there is a distribution of messages M that can be easily sampled using a t-time randomized algorithm such that the distribution of $H(M)$ is γ-uniform in $[0, n-1]$. It is reasonable to assume that common cryptographic hash functions such as SHA-1 are (t, γ)-uniform for some $\gamma \leq 2$ and some small value of t.

Security of GenericDSA. We are now ready to state the result of GenericDSA in the generic-group model. Let $\mathbb{G}$ be some prime-order group of order n (then that $\mathbb{G}$ is isomorphic to $\mathbb{Z}_n$). Let S be the canonical set of binary strings used to represent the elements of $\mathbb{G}$. For example, if $\mathbb{G} = E/\mathbb{F}_p$ then S is made up of pairs of integers in $[0, p-1]$. Let $F : \mathbb{G} \to [0, n-1]$ be a (t_1, α)-almost-invertible reduction function. Let $H : \{0,1\}^* \to [0, n-1]$ be a (t_2, γ)-uniform hash function.

Theorem 1 (Brown) *Using the notation above, if there exists a (t, ϵ, q, q_G) forger for GenericDSA in the generic-group model then there exists a (t', ϵ') collision finder for H where*

$$\epsilon' \geq \epsilon e^{-\alpha-\gamma} - 3\binom{q+q_G}{2}/n \qquad \text{and} \qquad t' \leq t + C((q+q_G)(t_1+t_2) + (\log n)(q+q_G))$$

for some absolute constant C that depends only on details of the model of computation.

The theorem shows that if H is collision resistant then GenericDSA is secure in the generic-group model. We plug-in some numbers to estimate the resulting security of GenericDSA. Suppose $n \approx 2^{160}$. When using SHA-1 there is no known algorithm for finding a collision with probability $\epsilon' \approx 1$ in time $t' \approx 2^{78}$. To get a signature forgery with probability $\epsilon \approx 1$ we must have $q + q_G \ll 2^{80}$. This means that, assuming SHA-1 is collision resistant, any generic-group forging algorithm FG will have running time at least

$$t \geq t' - C((q+q_G)(t_1+t_2) + (\log n)(q+q_G)) \approx 2^{78}$$

This is adequate security for a signature scheme.

Theorem 1 applies to ECDSA in exactly the same way. Note that for proper security we need to make sure that the reduction function $F : E/\mathbb{F}_q \to [0, n-1]$ is α-almost-invertible for a relatively small value of α, e.g. $\alpha < 10$.

7 Implementation Attacks on ECDSA

The analysis so far focused on the security of the ECDSA specification. Several recent results show that one must be particularly careful when implementing the pseudorandom generator used in ECDSA. Recall that in Step 1 of the signing algorithm we pick a random $k \in [1, n-1]$. For proper security of ECDSA it is critical that k be independently and uniformly distributed in $[1, n-1]$.

Howgrave-Graham and Smart [9] recently showed that for DSA if one can obtain the 8 most significant bits of k (out of the 160-bits of k) for 30 signatures then one can recover the private signing key in about 8 seconds just given the public key, the 30 signatures, and the thirty 8-bits out of k. These 8-bits out of k can become known to an attacker as a result of a weak pseudorandom generator. Recently, Bleichenbacher [4] was able to extend the results of [9] and further reduce the amount of information needed about k. It is not known whether these results generalize to ECDSA, but the lesson here is clear: the value of k must be independently and uniformly distributed in $[1, n-1]$ and must be kept secret so that an attacker cannot learn any information about k. Otherwise, not only will a given signature be untrustworthy, but the entire signature key may be compromised.

An earlier result of Bellare et al. [3] shows that if one uses a weak pseudorandom generator to generate k in DSA then a few signatures can leak the entire private key. Specifically, they show

that if one uses a linear congruential generator to generate k in DSA then the private signing key can be recovered using a constant number of consecutive signatures. We have not investigated whether or not these results generalize to ECDSA. Again, the lesson is that k must be generated using a strong pseudorandom generator. The values of k in consecutive applications of the signing algorithm should be indistinguishable to the adversary from uniform independent values.

8 The use of Koblitz Curves in ECDSA

In this section we study the security of Koblitz curves that are used for speeding up the arithmetic operations needed for various elliptic curve cryptosystems.

Koblitz curves [11] (also known as anomalous binary curves) are elliptic curves over $\mathbb{F}_{2^m}$ that have coefficients in $\mathbb{F}_2$. Koblitz curves are defined by one of the following two equations:

$$y^2 + xy = x^3 + 1 \qquad \text{or} \qquad y^2 + xy = x^3 + x^2 + 1$$

These curves are well suited for ECDSA since they speed-up both signature generation and signature verification.

The question is how secure is the discrete-log problem on these curves. The best discrete-log algorithm for general elliptic curves is the parallel collision search based on the Pollard Rho method [16]. The expected number of iterations to solve the discrete-log problem using this method is $\sqrt{\pi n/2} = O(\sqrt{n})$ where n is the number of points on the curve. When the order of the curve is $n \approx 2^{160}$ the time to compute discrete-log using this method is approximately 2^{80}. This offers adequate security by today's standards.

The question is whether better methods exist for Koblitz curves. Recently, Wiener and Zuccherato [17] and Gallant, Lambert, and Vanstone [7] showed that the extra structure of Koblitz curves which makes them useful for fast implementations can also be used to slightly speed-up the parallel collision-search method. They show that for Koblitz curves over $\mathbb{F}_{2^m}$ the parallel collision-search method can be sped up by a factor of $\sqrt{2m}$. Hence, discrete-log on Koblitz curves over $\mathbb{F}_{2^m}$ can be solved in time $\sqrt{\pi 2^m/4m}$. When $m \approx 160$ (e.g. $m = 163$) the time to compute discrete-log becomes approximately 2^{76}, a speed-up by a factor of 16. This level of security is adequate for deployed systems—this discrete log problem would take $5.7 \cdot 10^{14}$ years on a single machine running 100 iterations per second.

The small improvement in the running-time of Pollard rho for computing discrete-log on Koblitz curves is not a serious threat for using these curves. However, the extra structure on these curves may lead to new discrete-log algorithms for them. So far this has not happened. We note that the recent Weil decent techniques for computing discrete-log on curves over composite fields of low characteristic do not apply so far to Koblitz curves.

9 Corrections and Criticisms to [5]

We list some comments about [5]. These range from trivial typos to reasonably signficant gaps.

1. p. 3, Definition 1, line 5, there is an extra "," after "private key".
2. p. 3. In the description of signature generation, step 1, k might as well be from $[1..n-1]$. In step 5, s must not be 0. If it is, go back to step 1.

3. p. 3. The derivation of ECDSA from GenericDSA given in the paper only applies to curves defined over a prime finite field. The author should briefly describe the derivation for curves over fields of characteristic 2.

4. p. 3, Definition 2. B must be a finite set.

5. p. 3, Definition 2. "efficient" is not defined, and the non-asymptotic setting that this paper works in makes it unclear what is the intent.

6. p. 3, Definition 2. "the resulting distribution ... is indistinguishable from the uniform distribution". It is not clear what indistinguishable from the uniform distribution mean here. To make the proofs go through the author needs the notion of α-uniform distributions defined in Section 6 of this report.

7. Top of p. 4. The author argues that for ECDSA the reduction function f is almost invertible. It does not seem true that the resulting distribution is indistinguishable from uniform.

8. p. 4, Definition 3. It would be better to parameterize "time" as two components: actual running time and number of queries to the group oracle. Merging those together doesn't make conceptual sense.

9. p. 5, Definition 5 and footnote 2. I appreciate your trying to sidestep this issue of an efficient collision-finder existing even if one is not known, but ultimately I don't think that anything that flows from this can be rigorous. Consider the first full paragraph on p. 5 where you speak of SHA-1 being $(\epsilon, 2^{80})$-collision-resistant. Ultimately this just doesn't make any sense.

10. p. 5, Definition 6. Again, *easily sampled* is not not defined; and *indistinguishable from the uniform distribution* is not defined. Moreover, neither of these can easily be made rigorous with the *non-asymptotic* framework that you are using.

11. p. 5, first paragraph following Definition 6, *a constant hash function ... may be one-way under our definitions.* Certainly not; one-way-ness is defined on p. 4 and an inverting algorithm that outputs any point in $\{0,1\}^*$ is breaking the one-way-ness of the hash function $h(x) = c$.

12. p. 6, paragraph beginning *It may.* I found this quite imprecise; what does it mean that an adversary *can select an arbitrary Q_i in S*? Are you embellishing the model of computation somehow? What *is* the set S you are choosing for your results? Who chooses S? Our attempt to clarify these points is given in the body of this report.

13. p. 6, Lemma 1, the word *the* isn't quite justified in *the* combination, as more than one is possible. (But the proof subsequent proof names a unique one.)

14. p. 7, line 5, missing word *the*. Line 7, seems like an extra *s* at end of *unknowns values* (but then I see you use this again).

15. p. 7, line 2 of paragraph beginning "The function", missing article *a* before "source".

16. p. 7, same paragraph, sentence saying $O(n)$ memory is agrammatical and, what's more, you mean $\Omega(n)$ memory.

17. p. 7, last paragraph might begin "Informally," or some such thing.

18. p. 8, Lemma 3, typo, *and observer*.

19. p. 8, one-sentence paragraph following proof of lemma 3 should be reworded.

20. p. 8, final paragraph. I'm not sure I particularly buy the contents of this paragraph. Regardless, it is rather misplaced here in the text. Also, I believe you mean $\Theta(2^{k/2})$ on line last−3.

21. p. 8. In this same paragraph, the authors refer to the random-oracle model and the generic-group model as "assumptions". These are not complexity assumptions; they are models.

22. p. 9, Fact 1. A more precise statement would be desirable.

23. p. 11, p. 13–16. Proof of Theorem 2. When generating responses to group oracle queries the simulator has to make sure there are no collisions—i.e., distinct elements mapping to the same string in S. This seems to be completely ignored in the proof, and it impacts the bound. In general, though the simulation seem reasonable, its analysis is very sketch and seems to omit signficant terms.

24. p. 11, p. 16. Proof of Theorem 2. The author refers the reader to the appendix for a full proof of the theorem. However, the appendix contains only a very sketchy proof. This is a shame considering that Theorem 2 is the main result of the paper. The author really should write a full proof for Theorem 2, not only constructing the collision-finding algorithm, but carefully analyzing its success probability, too.

25. p. 11, Equation 10. What if the resulting R_{3j} happens to fall in $\{R_1, \ldots, R_{3j-1}\}$? This event needs to be taken into account in the failure probability of the simulation.

26. p. 11. third line below equation 10. Again, not clear what 'indistinguishable from the uniform distribution over S' means.

27. p. 12, line 1, eliminate word "call".

10 Conclusions

The ECDSA is an efficient and sensible algorithm, suitable for standardization. The proof in [5] for security of ECDSA has problems, but it appears that these problems can be cleaned up. A (corrected) proof in this domain provides signficant assurance, though that assurance is certainly less than that of a standard-model proof of existential unforgeability. The concrete security bounds are acceptable. An ECDSA standard should give clear implementation advice about acceptable curves, and caveats about the source of random numbers used.

References

[1] **ANSI X9.62**. Public key cryptography for the financial services industry: the elliptic curve digital signature algorithm (ECDSA). American National Standards Institute, 1999.

[2] **M. Bellare** and **P. Rogaway**. Random oracles are practical: A paradigm for designing efficient protocols. First ACM Conference on Computer and Communications Security, ACM, 1993.

[3] **M. Bellare**, **S. Goldwasser**, and **D. Micciancio**. Pseudo-Random Number Generation within Cryptographic Algorithms: the DSS Case. Proc. of Crypto '97, LNCS 1294, 1997.

[4] **D. Bleichenbacher**. Private communication.

[5] **D. Brown**. The exact security of ECDSA. Technical Report CORR 2000-54, Dept. of Combinatorics and Optimization, University of Waterloo, 2000. Available from www.cacr.math.uwaterloo.ca

[6] **R. Canetti**, **O. Goldreich**, and **S. Halevi**. The random oracle model revisited, in proc. STOC '98.

[7] **R. Gallant**, **R. Lambert**, and **S. Vanstone**. Improving the parallelized Pollard lambda search on binary anomalous curves, manuscript.

[8] **S. Goldwasser**, **S. Micali**, and **R. Rivest**. A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal of Computing, vol. 17, pp. 281–308, 1988.

[9] **N. Howgrave-Graham** and **N. Smart**. Lattice attacks on digital signature schemes, manuscript.

[10] **IEEE 1363-2000**. Standard specifications for public key cryptography. Institute of Electrical and Electronics Engineers, 2000.

[11] **N. Koblitz**. CM-curves with good cryptographic properties, Proc. Crypto '91, LNCS 576, pp. 279–287, 1991.

[12] **U. Maurer** and **S. Wolf**. Lower bounds on generic algorithms in groups. Advances in Cryptology — EUROCRYPT '98. Lecture Notes in Computer Science, vol. 1403, Springer-Verlag, 1998.

[13] **V. Nechaev**. Complexity of a determinate algorithm for the discrete logarithm. Mathematical Notes, vol. 55, no. 2, pp. 165–172, 1994.

[14] **FIPS 186-2**. Digital signature Standard. National Institute of Standards and Technology, 2000.

[15] **V. Shoup**. Lower bounds for discrete logarithms and related problems. Advances in Cryptology — EUROCRYPT '97. Lecture Notes in Computer Sciences, vol. 1233, Springer-Verlag, pp. 256–266, 1997.

[16] **P. Van Oorschot** and **M. Wiener**. Parallel collision search with cryptanalytic applications, J. of Cryptology, Vol. 12, No. 1, 1999.

[17] **M. Wiener** and **R. Zuccherato**. Faster attacks on elliptic curve cryptosystems, manuscript.