Evaluation of Security Level of Cryptography: ECDSA Signature Scheme

Alfred Menezes, Minghua Qu, Doug Stinson, Yongge Wang
Certicom Research
Contact: amenezes@certicom.com

January 15, 2001

1 Summary

This document summarizes the state-of-the-art in knowledge on the security of the elliptic curve digital signature algorithm (ECDSA). ECDSA is the elliptic curve analogue of the digital signature algorithm (DSA) and has been standardized by many standards organizations around the world including NIST, IEEE, ANSI and ISO. The security of DSA, ECDSA and related discrete logarithm signature schemes have been widely studied by the cryptographic community. Our study found no new weaknesses in ECDSA.

2 Protocol specification

2.1 EC domain parameters

Elliptic Curve domain parameters are comprised of:

1. a field size q, where either $q = p$, an odd prime, or $q = 2^m$;
2. an indication FR (*field representation*) of the representation used for the elements of $\mathbb{F}_q$;
3. (optional) a bit string `seedE` of length at least 160 bits, if the elliptic curve was generated in accordance with the method described in the submission;
4. two field elements a and b in $\mathbb{F}_q$ which define the equation of the elliptic curve E over $\mathbb{F}_q$ (i.e., $y^2 = x^3 + ax + b$ in the case $p > 3$, and $y^2 + xy = x^3 + ax^2 + b$ in the case $q = 2^m$);
5. two field elements x_G and y_G in $\mathbb{F}_q$ which define a finite point $G = (x_G, y_G)$ of prime order in $E(\mathbb{F}_q)$;
6. the order n of the point G, with $n > 2^{160}$ and $n > 4\sqrt{q}$; and
7. the cofactor $h = \#E(\mathbb{F}_q)/n$.

2.2 ECDSA key pairs

An entity A's key pair is associated with a particular set of EC domain parameters $D = (q, \text{FR}, a, b, G, n, h)$. This association can be assured cryptographically (e.g., with certificates) or by context (e.g., all entities use the same domain parameters). The entity A must have the assurance that the domain parameters are valid (see the submission) prior to key generation.

ECDSA KEY PAIR GENERATION. Each entity A does the following:

1. Select a random or pseudorandom integer d in the interval $[1, n-1]$.
2. Compute $Q = dG$.
3. A's public key is Q; A's private key is d.

2.3 ECDSA signature generation

To sign a message m, an entity A with domain parameters $D = (q, \mathrm{FR}, a, b, G, n, h)$ and associated key pair (d, Q) does the following:

1. Select a random or pseudorandom integer k, $1 \leq k \leq n-1$.
2. Compute $kG = (x_1, y_1)$ and convert x_1 to an integer $\overline{x}_1$.
3. Compute $r = \overline{x}_1 \bmod n$. If $r = 0$ then go to step 1.
4. Compute $k^{-1} \bmod n$.
5. Compute SHA-1(m) and convert this bit string to an integer e.
6. Compute $s = k^{-1}(e + dr) \bmod n$. If $s = 0$ then go to step 1.
7. A's signature for the message m is (r, s).

2.4 ECDSA signature verification

To verify A's signature (r, s) on m, B obtains an authentic copy of A's domain parameters $D = (q, \mathrm{FR}, a, b, G, n, h)$ and associated public key Q. It is recommended that B also validates D and Q (see the submission). B then does the following:

1. Verify that r and s are integers in the interval $[1, n-1]$.
2. Compute SHA-1(m) and convert this bit string to an integer e.
3. Compute $w = s^{-1} \bmod n$.
4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $X = u_1 G + u_2 Q$.
6. If $X = O$, then reject the signature. Otherwise, convert the x-coordinate x_1 of X to an integer $\overline{x}_1$, and compute $v = \overline{x}_1 \bmod n$.
7. Accept the signature if and only if $v = r$.

PROOF THAT SIGNATURE VERIFICATION WORKS. If a signature (r,s) on a message m was indeed generated by A, then $s = k^{-1}(e+dr) \bmod n$. Rearranging gives

$$k \equiv s^{-1}(e+dr) \equiv s^{-1}e + s^{-1}rd \equiv we + wrd \equiv u_1 + u_2 d \pmod{n}.$$

Thus $u_1G + u_2Q = (u_1 + u_2 d)G = kG$, and so $v = r$ as required.

3 Security level of cryptographic techniques

The security objective of ECDSA is to be existentially unforgeable against a chosen-message attack. The goal of an adversary who launches such an attack against a legitimate entity A is to obtain a valid signature on a single message m, after having obtained A's signature on a collection of messages (not including m) of the adversary's choice.

Some progress has been made on proving the security of ECDSA, albeit in strong theoretical models. Slight variants of DSA and ECDSA (but not ECDSA itself) have been proven to be existentially unforgeable against chosen-message attack by Pointcheval and Stern [32] (see also [7]) under the assumptions that the discrete logarithm problem is hard and that the hash function employed is a random function. Specifically, their results show that any successful attack which does not use the structure of the specific hash function in the scheme must break the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDSA itself has been recently proven secure by Brown [8] under the strong assumption that the underlying group is a generic group and that the hash function employed is collision resistant. Specifically, Brown's results show that any successful attack which does not use the structure of the Elliptic Curve group must find collisions in the hash function.

4 Security level of cryptographic primitive functions

4.1 Attacks on the hash function

DEFINITION. A *(cryptographic) hash function* H is a function that maps bit strings of arbitrary lengths to bit strings of a fixed length t such that:

1. H can be computed efficiently;
2. (*preimage resistance*) For y selected uniformly at random from $\{0,1\}^t$, it is computationally infeasible to find a bit string x such that $H(x) = y$.
3. (*second preimage resistance*) Given x_1, it is computationally infeasible to find a different bit string x_2 such that $H(x_1) = H(x_2)$.

4. (*collision resistance*) It is computationally infeasible to find distinct bit strings x_1 and x_2 such that $H(x_1) = H(x_2)$.

SHA-1 SECURITY REQUIREMENTS. The following gives some examples of how attacks on ECDSA can be successfully launched if SHA-1 is not preimage resistant, second preimage resistant, or collision resistant.

1. If SHA-1 is not preimage resistant, then an adversary E may be able to forge A's signatures as follows. E selects an arbitrary integer l, and computes r as the x-coordinate of $Q + lG$ reduced modulo n. E sets $s = r$ and computes $e = rl \bmod n$. If E can find a message m such that $e = \text{SHA-1}(m)$, then (r,s) is a valid signature for m.

2. If SHA-1 is not collision resistant, then an entity A may be able to repudiate signatures as follows. A first generate two messages m and m' such that $\text{SHA-1}(m) = \text{SHA-1}(m')$; such a pair of messages is called a *collision* for SHA-1. She then signs m, and later claims to have signed m' (note that every signature for m is also a signature for m').

3. If SHA-1 is not collision resistant, then an entity B could use the birthday attack to swindle A as follows. B prepares two versions (m_1 and m_2) of a contract, where m_1 is favorable to A and m_2 bankrupts A. B makes several subtle changes to each document and compares whether $\text{SHA-1}(m_1') = \text{SHA-1}(m_2')$, where m_1' (m_2') is a subtle variant of m_1 (m_2). When B finds two such variants, B gets the signature (r,s) of m_1' from A. Then (r,s) is also a signature of m_2'.

4. If SHA-1 is not second preimage resistant, then an entity B may be able to forge signatures as follows. B generates a message m, find another message m' such that $\text{SHA-1}(m) = \text{SHA-1}(m')$, and gets the signature (r,s) of m' from A. Then (r,s) is also a signature of m.

IDEAL SECURITY. A t-bit hash function is said to be have *ideal security* [26] if both: (i) given a hash output, producing a preimage (or a second preimage) requires approximately 2^t operations; and (ii) producing a collision requires approximately $2^{t/2}$ operations (this is the best that can be hoped for, in view of the birthday attacks). SHA-1 is a 160-bit hash function and is believed to have ideal security. The fastest method known for attacking ECDSA by exploiting properties of SHA-1 is to find collisions for SHA-1. Since this is believed to take 2^{80} steps, attacking ECDSA in this way is computationally infeasible. Note, however, that this attack imposes an upper bound of 2^{80} on the security level of ECDSA, regardless of the size of the primary security parameter n. Of course, this is also the case with all present signature schemes with appendix since the only hash functions that are widely accepted as being both secure and practical are SHA-1 and RIPEMD-160 (see Dobbertin, Bosselaers and Preneel [10]), both of which are 160-bit hash functions.

VARIABLE OUTPUT LENGTH HASH FUNCTIONS. It is expected that SHA-1 will soon be replaced by a family of hash functions H_l, where H_l is an l-bit hash function having ideal

security. If one uses ECDSA with parameter n, then one would use H_l, where $l = \lfloor \log_2 n \rfloor$, as the hash function. In this case, attacking ECDSA by solving the ECDLP and attacking ECDSA by finding collisions for H_l, both take approximately the same amount of time. The new family will have output lengths of 256, 384 and 512 bits [29].

VAUDENAY'S ATTACKS. Vaudenay [43] demonstrated a theoretical weakness in DSA based on his insight that the actual hash function used in the DSA is SHA-1 modulo q, not just SHA-1, where q is a 160-bit prime. (Since SHA-1 is a 160-bit hash function, some of its outputs, when converted to integers, are larger than q. Hence, in general, SHA-1$(m) \neq$ (SHA-1(m) mod q).) This weakness allows the selective forgery of one message if the adversary can select the domain parameters. This weakness is not present in ECDSA because of the requirement that n (the analogous quantity to q in the DSA) be greater than 2^{160}.

5 Security level of cryptographic primitive problem: the elliptic curve discrete logarithm problem

One way in which an adversary can succeed is to compute A's private key d from A's domain parameters $(q, \mathrm{FR}, a, b, G, n, h)$ and public key Q. Then the adversary could subsequently forge A's signature on any message of its choice.

PROBLEM DEFINITION. The *elliptic curve discrete logarithm problem (ECDLP)* is the following: given an elliptic curve E defined over a finite field $\mathbb{F}_q$, a point $P \in E(\mathbb{F}_q)$ of order n, and a point $Q = lP$ where $0 \leq l \leq n-1$, determine l.

5.1 Known attacks

This subsection overviews the algorithms known for solving the ECDLP and discusses how they can be avoided in practice.

1. NAIVE EXHAUSTIVE SEARCH. In this method, one simply computes successive multiples of P: P, $2P$, $3P$, $4P, \ldots$ until Q is obtained. This method can take up to n steps in the worst case.

2. POHLIG-HELLMAN ALGORITHM. This algorithm, due to Pohlig and Hellman [31], exploits the factorization of n, the order of the point P. The algorithm reduces the problem of recovering l to the problem of recovering l modulo each of the prime factors of n; the desired number l can then be recovered by using the Chinese Remainder Theorem.

   The implications of this algorithm are the following. To construct the most difficult instance of the ECDLP, one must select an elliptic curve whose order is divisible by a large

prime n. Preferably, this order should be a prime or almost a prime (i.e. a large prime n times a small integer h). For the remainder of this section, we shall assume that the order n of P is prime.

3. BABY-STEP GIANT-STEP ALGORITHM. This algorithm is a time-memory trade-off of the method of exhaustive search. It requires storage for about $\sqrt{n}$ points, and its running time is roughly $\sqrt{n}$ steps in the worst case.

4. POLLARD'S RHO ALGORITHM. This algorithm, due to Pollard [33], is a randomized version of the baby-step giant-step algorithm. It has roughly the same expected running time ($\sqrt{\pi n/2}$ steps) as the baby-step giant-step algorithm, but is superior in that it requires a negligible amount of storage.

   Gallant, Lambert and Vanstone [16], and Wiener and Zuccherato [44] showed how Pollard's rho algorithm can be sped up by a factor of $\sqrt{2}$. Thus the expected running time of Pollard's rho method with this speedup is $(\sqrt{\pi n})/2$ steps.

5. PARALLELIZED POLLARD'S RHO ALGORITHM. Van Oorschot and Wiener [30] showed how Pollard's rho algorithm can be parallelized so that when the algorithm is run in parallel on r processors, the expected running time of the algorithm is roughly $(\sqrt{\pi n})/(2r)$ steps. That is, using r processors results in an r-fold speed-up.

6. POLLARD'S LAMBDA METHOD. This is another randomized algorithm due to Pollard [33]. Like Pollard's rho method, the lambda method can also be parallelized with a linear speedup. The parallelized lambda-method is slightly slower than the parallelized rho-method [30]. The lambda-method is, however, faster in situations when the logarithm being sought is known to lie in a subinterval $[0,b]$ of $[0,n-1]$, where $b < 0.39n$ [30].

7. MULTIPLE LOGARITHMS. R. Silverman and Stapleton [37] observed that if a single instance of the ECDLP (for a given elliptic curve E and base point P) is solved using (parallelized) Pollard's rho method, then the work done in solving this instance can be used to speed up the solution of other instances of the ECDLP (for the same curve E and base point P). More precisely, if the first instance takes expected time t, then the second instance takes expected time $(\sqrt{2}-1)t \approx 0.41t$. Having solved these two instances, the third instance takes expected time $(\sqrt{3}-\sqrt{2})t \approx 0.32t$. Having solved these three instances, the fourth instance takes expected time $(\sqrt{4}-\sqrt{3})t \approx 0.27t$. And so on. Thus subsequent instances of the ECDLP for a particular elliptic curve become progressively easier. Another way of looking at this is that solving k instances of the ECDLP (for the same curve E and base point P) takes only $\sqrt{k}$ as much work as it does to solve one instance of the ECDLP. This analysis does not take into account storage requirements.

   Concerns that successive logarithms become easier can be addressed by ensuring that the elliptic parameters are chosen so that the first instance is infeasible to solve.

8. SUPERSINGULAR ELLIPTIC CURVES. Menezes, Okamoto and Vanstone [25, 24] and Frey and Rück [14] showed how, under mild assumptions, the ECDLP in an elliptic curve E defined over a finite field $\mathbb{F}_q$ can be reduced to the ordinary DLP in the multiplicative group of some extension field $\mathbb{F}_{q^k}$ for some $k \geq 1$, where the number field sieve algorithm applies. The reduction algorithm is only practical if k is small — this is not the case for most elliptic curves, as shown by Balasubramanian and Koblitz [4]. To ensure that the reduction algorithm does not apply to a particular curve, one only needs to check that n, the order of the point P, does not divide $q^k - 1$ for all small k for which the DLP in $\mathbb{F}_{q^k}$ is tractable — in practice, when $n > 2^{160}$ then $1 \leq k \leq 20$ suffices [2].

   An elliptic curve E over $\mathbb{F}_q$ is said to be *supersingular* if the trace t of E is divisible by the characteristic p of $\mathbb{F}_q$. For this very special class of elliptic curves, it is known that $k \leq 6$. It follows that the reduction algorithm yields a subexponential-time algorithm for the ECDLP in supersingular curves. For this reason, supersingular curves are explicitly excluded from use in the ECDSA by the above divisibility check.

   More generally, the divisibility check rules out all elliptic curves for which the ECDLP can be efficiently reduced to the DLP in some small extension of $\mathbb{F}_q$. These include the supersingular elliptic curves and elliptic curves of trace 2 (elliptic curves E over $\mathbb{F}_q$ for which $\#E(\mathbb{F}_q) = q - 1$).

9. PRIME-FIELD ANOMALOUS CURVES. An elliptic curve E over $\mathbb{F}_p$ is said to be *prime-field-anomalous* if $\#E(\mathbb{F}_p) = p$. Semaev [35], Smart [39], and Satoh and Araki [34] showed how to efficiently solve the ECDLP for these curves. The attack does not extend to any other classes of elliptic curves. Consequently, by verifying that the number of points on an elliptic curve is not equal to the cardinality of the underlying field, one can easily ensure that the Semaev-Smart-Satoh-Araki attack does not apply.

10. CURVES DEFINED OVER A SMALL FIELD. Suppose that E is an elliptic curve defined over the finite field $\mathbb{F}_{2^e}$. Gallant, Lambert and Vanstone [16], and Wiener and Zuccherato [44] showed how Pollard's rho algorithm for computing elliptic curve logarithms in $E(\mathbb{F}_{2^{ed}})$ can be further sped up by a factor of $\sqrt{d}$ — thus the expected running time of Pollard's rho method for these curves is $(\sqrt{\pi n/d})/2$ steps. For example, if E is a Koblitz curve (see the submission), then Pollard's rho algorithm for computing elliptic curve logarithms in $E(\mathbb{F}_{2^m})$ can be sped up by a factor of $\sqrt{m}$. This speedup should be considered when doing a security analysis of elliptic curves whose coefficients lie in a small subfield.

11. CURVES DEFINED OVER $\mathbb{F}_{2^m}$, m COMPOSITE. Galbraith and Smart [15], expanding on earlier work of Frey [12, 13], discuss how the Weil descent might be used to solve the ECDLP for elliptic curves defined over $\mathbb{F}_{2^m}$ where m is composite (such fields are sometimes called *composite* fields). More recently, Gaudry, Hess and Smart [17] refined these ideas to provide some evidence that when m has a small divisor l, e.g. $l = 4$, the ECDLP for elliptic curves defined over $\mathbb{F}_{2^m}$ can be solved faster than with Pollard's rho

algorithm. See also Menezes and Qu [27] for an analysis of the Weil descent attack. In light of these results, it seems prudent to not use elliptic curves over composite fields.

It should be noted that some ECC standards, including the draft ANSI X9.63 [3], explicitly exclude the use of elliptic curves over composite fields. The ANSI X9F1 committee also agreed in January 1999 to exclude the use of such curves in a forthcoming revision of ANSI X9.62.

12. NON-APPLICABILITY OF INDEX-CALCULUS METHODS. Whether or not there exists a general subexponential-time algorithm for the ECDLP is an important unsettled question, and one of great relevance to the security of ECDSA. It is extremely unlikely that anyone will ever be able to *prove* that no subexponential-time algorithm exists for the ECDLP. However, much work has been done on the DLP over the past 24 years, and more specifically on the ECDLP over the past 16 years, and no subexponential-time algorithm has been discovered for the ECDLP. Miller [28] and J. Silverman and Suzuki [38] have given convincing arguments for why the most natural way in which the index-calculus algorithms can be applied to the ECDLP is most likely to fail.

13. XEDNI-CALCULUS ATTACKS. A very interesting line of attack on the ECDLP, called the *xedni-calculus attack* was recently proposed by J. Silverman [36]. One intriguing aspect of the xedni-calculus is that it can be adapted to solve both the ordinary discrete logarithm and the integer factorization problems. However, it was subsequently shown by a team of researchers including J. Silverman (see Jacobson et al. [19]) that the attack is virtually certain to fail in practice.

14. HYPERELLIPTIC CURVES. Hyperelliptic curves are a family of algebraic curves of arbitrary genus that includes elliptic curves. Hence, an elliptic curve can be viewed as a hyperelliptic curve of genus 1. Adleman, DeMarrais and Huang [1] (see also Stein, Müller and Thiel [41]) presented a subexponential-time algorithm for the discrete logarithm problem in the jacobian of a large genus hyperelliptic curve over a finite field. However, in the case of elliptic curves, the algorithm is worse than naive exhaustive search.

15. EQUIVALENCE TO OTHER DISCRETE LOGARITHM PROBLEMS. Stein [40] and Zuccherato [45] showed that the discrete logarithm problem in real quadratic congruence function fields of genus 1 is equivalent to the ECDLP. Since no subexponential-time algorithm is known for the former problem, this may provide further evidence for the hardness of the ECDLP.

5.2 Experimental results

The best general-purpose algorithm known for the ECDLP is the parallelized version of Pollard's rho algorithm which has an expected running time of $(\sqrt{\pi n})/(2r)$ steps, where n is the (prime) order of the base point P, and r is the number of processors utilized.

CERTICOM'S ECC CHALLENGE. Certicom initiated an ECC challenge [9] in November 1997 in order to encourage and stimulate research on the ECDLP. Their challenges consist of instances of the ECDLP on a selection of elliptic curves. The challenge curves are divided into three categories listed below. In the following, ECCp-k denotes a random curve over a field $\mathbb{F}_p$, ECC2-k denotes a random curve over a field $\mathbb{F}_{2^m}$, and ECC2K-k denotes a Koblitz curve (see the submission) over $\mathbb{F}_{2^m}$; k is the bitlength of n. In all cases, the bitsize of the order of the underlying finite field is equal or slightly greater than k (so curves have either prime order or almost prime order).

1. Randomly generated curves over $\mathbb{F}_p$, where p is prime: ECCp-79, ECCp-89, ECCp-97, ECCp-109, ECCp-131, ECCp-163, ECCp-191, ECCp-239, and ECCp-359.
2. Randomly generated curves over $\mathbb{F}_{2^m}$, where m is prime: ECC2-79, ECC2-89, ECC2-97, ECC2-109, ECC2-131, ECC2-163, ECC2-191, ECC2-238, and ECC2-353.
3. Koblitz curves over $\mathbb{F}_{2^m}$, where m is prime: ECC2K-95, ECC2-108, ECC2-130, ECC2-163, ECC2-238, and ECC2-358.

RESULTS OF THE CHALLENGE. Escott et al. [11] report on their 1998 implementation of the parallelized Pollard's rho algorithm which incorporates some improvements of Teske [42]. The hardest instance of the ECDLP they solved was the Certicom ECCp-97 challenge. For this task they utilized over 1200 machines from at least 16 countries, and found the answer in 53 days. The total number of steps executed was about 2×10^{14} elliptic curve additions which is close to the expected time ($(\sqrt{\pi n})/2 \approx 3.5 \times 10^{14}$, where $n \approx 2^{97}$). Escott et al. [11] conclude that the running time of Pollard's rho algorithm in practice fits well with the theoretical predictions. They estimate that the ECCp-109 challenge could be solved by a network of 50,000 Pentium Pro 200MHz machines in about 3 months.

5.3 Hardware attacks

Van Oorschot and Wiener [30] examined the feasibility of implementing parallelized Pollard's rho algorithm using special-purpose hardware. They estimated that if $n \approx 10^{36} \approx 2^{120}$, then a machine with $r = 330,000$ processors could be built for about US $10 million that could compute a single elliptic curve discrete logarithm in about 32 days. Since ANSI X9.62 mandates that the parameter n should satisfy $n > 2^{160}$, such hardware attacks appear to be infeasible with today's technology.

6 Other attacks

IMPLEMENTATION ATTACKS. The existing ECDSA standards do not address attacks that could be launched against implementations of ECDSA such as timing attacks (Kocher [21]), differential fault analysis (Boneh, DeMillo and Lipton [6]), differential power analysis (Kocher, Jaffe and Jun [22]), and attacks which exploit weak pseudorandom number generators (Kelsey et al. [20]).

SECURITY REQUIREMENTS FOR PER-MESSAGE SECRETS. The per-message secrets k in ECDSA signature generation have the same security requirements as the private key d. This is because if an adversary E learns a single per-message secret k which was used by A to generate a signature (r,s) on some message m, then E can recover A's private key since $d = r^{-1}(ks - e) \bmod n$ where $e = \text{SHA-1}(m)$ (see step 6 of ECDSA signature generation). Hence per-message secrets must be securely generated, securely stored, and securely destroyed after they have been used.

REPEATED USE OF PER-MESSAGE SECRETS. The per-message secrets k used to sign two or more messages should be generated independently of each other. In particular, a different per-message secret k should be generated for each different message signed; otherwise, the private key d can be recovered. Note that if a secure random or pseudorandom number generator is used, then the chance of generating a repeated k value is negligible. To see how private keys can be recovered if per-message secrets are repeated, suppose that the same per-message secret k was used to generate ECDSA signatures (r,s_1) and (r,s_2) on two different messages m_1 and m_2. Then $s_1 \equiv k^{-1}(e_1 + dr) \pmod n$ and $s_2 \equiv k^{-1}(e_2 + dr) \pmod n$, where $e_1 = \text{SHA-1}(m_1)$ and $e_2 = \text{SHA-1}(m_2)$. Then $ks_1 \equiv e_1 + dr \pmod n$ and $ks_2 \equiv e_2 + dr \pmod n$. Subtraction gives $k(s_1 - s_2) \equiv e_1 - e_2 \pmod n$. If $s_1 \not\equiv s_2 \pmod n$, which occurs with overwhelming probability, then $k \equiv (s_1 - s_2)^{-1}(e_1 - e_2) \pmod n$. Thus, an adversary can determine k, and then use this to recover d, as described above.

ATTACK USING KNOWN RELATIONS ON PER-MESSAGE SECRETS. There should also be no obvious relations between the per-message secrets. For example, suppose that there is a relation $k_2 = ak_1 + b$ for some fixed a and b. Since $k_1 s_1 = e_1 + dr_1$ and $k_2 s_2 = e_2 + dr_2$, we can get two equations $k_1 s_1 = e_1 + dr_1$ and $(ak_1 + b)s_2 = e_2 + dr_2$ in two unknows k_1 and d. Hence one can solve the equations and obtain the value of d.

PARTIAL INFORMATION ON PER-MESSAGE SECRETS. In [18], Howgrave-Graham and Smart use lattice reduction techniques to show that partial information of DSA per-message secrets can lead to the complete exposure of the private key. For example, the attack is practical if a contiguous block of 5% of the bits in 20 per-message secrets is known, for p a 160 bit prime. Their results could easily be extended to ECDSA.

To avoid similar attacks where partial information about the per-message secrets may be leaked,

it is imperative that the per-message secrets be generated uniformly at random (or pseudorandom) from the interval $[1, n-1]$, and securely destroyed after usage.

DUPLICATE-SIGNATURE KEY SELECTION. A signature scheme S is said to have the duplicate-signature key selection (DSKS) property if given A's public key P_A and given A's signature s_A on a message M, an adversary E is able to select a valid key pair (P_E, S_E) for S such that s_A is also E's signature on M. Note that this definition requires that S_E is known to E. Blake-Wilson and Menezes [5] showed how this property can be exploited to attack a key agreement protocol which employs a signature scheme. They also demonstrated that if entities are permitted to select their own domain parameters, then ECDSA possesses the DSKS property. To see this, suppose that A's domain parameters are $D_A = (q, \mathrm{FR}, a, b, G, n, h)$, A's key pair is (Q_A, d_A), and (r, s) is A's signature on M. The adversary E selects an arbitrary integer c, $1 \leq c \leq n-1$, such that $t := ((s^{-1}e + s^{-1}rc) \bmod n) \neq 0$, computes $X = s^{-1}eG + s^{-1}rQ$ (where $e = \text{SHA-1}(M)$) and $\overline{G} = (t^{-1} \bmod n)X$. E then forms $D_E = (q, \mathrm{FR}, a, b, \overline{G}, n, h)$ and $Q_E = c\overline{G}$. Then it is easily verified that D_E and Q_E are valid, and that (r, s) is also E's signature on M.

If one mandates that the generating point G be selected verifiably at random (that is, verify that G is a valid point on the curve and it has the correct order) during domain parameter generation, then it appears that ECDSA no longer possesses the DSKS property. It must be emphasized that possession of the DSKS property does not constitute a weakness of the signature scheme — the goal of a signature scheme is to be existentially unforgeable against a chosen-message attack. Rather, it demonstrates the importance of auditing domain parameter and public key generation.

7 Recommended parameters

In Table 1, the security levels of the ECDSA with the parameters recommended in the submission are compared with other submitted schemes, with RSA, and with symmetric schemes (e.g., the Advanced Encryption Standard–AES). Some of the comparisons among symmetric key cryptography, RSA security, and ECDSA security are adapted from Lenstra and Verheul [23].

8 Performance comparison

Table 1 presented comparable key lengths of several schemes. Generally, the ESIGN signature scheme is faster than both ECDSA and RSA signature schemes with comparable key lengths. ACE, RSA, and ESIGN have roughly the same public key size for comparable security level. ACE and RSA have roughly the same private key size for comparable security level. ESIGN's private key size is roughly $2/3$ of the ACE (or RSA) private key size for comparable security level. ECDSA and MY-ELLTY have significantly smaller key size for comparable security

Table 1: Rough Comparison of Security Levels

Symmetric Key Size	RSA Modulus Size	ESIGN Modulus Size	ACE Modulus Size	ECDSA Key Size	MY-ELLTY Key Size
40					160
48					192
76	960	960	960	152	
80 (SKIPJACK)	1024	1024	1024	160	
112 (Triple-DES)	2048	2048	2048	224	
128 (128-bit AES)	3072	3072	3072	256	
192 (192-bit AES)	7680	7680	7680	384	
256 (256-bit AES)	15360	15360	15360	512	

level. However, the hash function used in MY-ELLTY is not collision resistant, thus it is not secure against attacks on the hash function.

References

[1] L. Adleman, J. DeMarrais and M. Huang, "A subexponential algorithm for discrete logarithms over the rational subgroup of the jacobians of large genus hyperelliptic curves over finite fields", *Algorithmic Number Theory*, Lecture Notes in Computer Science, **877** (1994), Springer-Verlag, 28-40.

[2] ANSI X9.62, *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, 1999.

[3] ANSI X9.63, *Public Key Cryptography for the Financial Services Industry: Elliptic Curve Key Agreement and Key Transport Protocols*, working draft, October 2000.

[4] R. Balasubramanian and N. Koblitz, "The improbability that an elliptic curve has subexponential discrete log problem under the Menezes–Okamoto–Vanstone algorithm", *Journal of Cryptology*, **11** (1998), 141-145.

[5] S. Blake-Wilson and A. Menezes, "Unknown key-share attacks on the station-to-station (STS) protocol", *Public Key Cryptography – Proceedings of PKC '99*, Lecture Notes in Computer Science, **1560** (1999), 154-170.

[6] D. Boneh, R. DeMillo and R. Lipton, "On the importance of checking cryptographic protocols for faults", *Advances in Cryptology – Eurocrypt '97*, Lecture Notes in Computer Science, **1233** (1997), Springer-Verlag, 37-51.

[7] E. Brickell, D. Pointcheval, S. Vaudenay and M. Yung, "Design validations for discrete logarithm based signature schemes", *Public Key Cryptography – Proceedings of PKC 2000*, Lecture Notes in Computer Science, **1751** (2000), 276-292.

[8] D. Brown, "The exact security of ECDSA", Technical report CORR 2000-54, Dept. of C&O, University of Waterloo, 2000. Available from http://www.cacr.math.uwaterloo.ca

[9] Certicom ECC Challenge, November 1997, http://www.certicom.com

[10] H. Dobbertin, A. Bosselaers and B. Preneel, "RIPEMD-160: A strengthened version of RIPEMD", *Fast Software Encryption – FSE '96*, Lecture Notes in Computer Science, **1039** (1996), Springer-Verlag, 71-82.

[11] A. Escott, J. Sager, A. Selkirk and D. Tsapakidis, "Attacking elliptic curve cryptosystems using the parallel Pollard rho method", *CryptoBytes – The Technical Newsletter of RSA Laboratories*, volume 4, number 2, Winter 1999, 15-19. Also available at http://www.rsasecurity.com

[12] G. Frey, "How to disguise an elliptic curve (Weil descent)", talk at ECC '98. Slides available at http://www.cacr.math.uwaterloo.ca

[13] G. Frey, "Applications of arithmetical geometry to cryptographic constructions", *Proceedings of the Fifth International Conference on Finite Fields and Applications*, to appear.

[14] G. Frey and H. Rück, "A remark concerning *m*-divisibility and the discrete logarithm in the divisor class group of curves", *Mathematics of Computation*, **62** (1994), 865-874.

[15] S. Galbraith and N. Smart, "A cryptographic application of Weil descent", *Codes and Cryptography*, Lecture Notes in Computer Science, **1746** (1999), Springer-Verlag, 191-200.

[16] R. Gallant, R. Lambert and S. Vanstone, "Improving the parallelized Pollard lambda search on binary anomalous curves", to appear in *Mathematics of Computation*.

[17] P. Gaudry, F. Hess and N. Smart, "Constructive and destructive facets of Weil descent on elliptic curves", preprint, January 2000. Available from http://www.hpl.hp.com/techreports/2000/HPL-2000-10.html

[18] N. Howgrave-Graham and N. Smart. Lattice attacks on digital signature schemes. To appear in: *Design, Codes, and Cryptography.*

[19] M. Jacobson, N. Koblitz, J. Silverman, A. Stein and E. Teske, "Analysis of the xedni calculus attack", *Designs, Codes and Cryptography*, **20** (2000), 41-64.

[20] J. Kelsey, B. Schneier, D. Wagner and C. Hall, "Cryptanalytic attacks on pseudorandom number generators", *Fast Software Encryption – FSE '98*, Lecture Notes in Computer Science, **1372** (1998), Springer-Verlag, 168-188.

[21] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems", *Advances in Cryptology – Crypto '96*, Lecture Notes in Computer Science, **1109** (1996), Springer-Verlag, 104-113.

[22] P. Kocher, J. Jaffe and B. Jun, "Differential power analysis", *Advances in Cryptology – Crypto '99*, Lecture Notes in Computer Science, **1666** (1999), Springer-Verlag, 388-397.

[23] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. Distributed in the *3rd workshop on Elliptic Curve Cryptography (ECC 99)*. Available from http://www.cryptosavvy.com/

[24] A. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, Boston, 1993.

[25] A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", *IEEE Transactions on Information Theory*, **39** (1993), 1639-1646.

[26] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[27] A. Menezes and M. Qu, "Analysis of the Weil descent attack of Gaudry, Hess and Smart", *Proceedings of RSA 2001*, 2001, to appear.

[28] V. Miller, "Uses of elliptic curves in cryptography", *Advances in Cryptology – Crypto '85*, Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 417-426.

[29] National Institute of Standards and Technology, "Descriptions of SHA-256, SHA-384, and SHA-512", preprint, 2000.

[30] P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", *Journal of Cryptology*, **12** (1999), 1-28.

[31] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance", *IEEE Transactions on Information Theory*, **24** (1978), 106-110.

[32] D. Pointcheval and J. Stern, "Security proofs for signature schemes", *Advances in Cryptology – Eurocrypt '96*, Lecture Notes in Computer Science, **1070** (1993), Springer-Verlag, 387-398.

[33] J. Pollard, "Monte Carlo methods for index computation mod p", *Mathematics of Computation*, **32** (1978), 918-924.

[34] T. Satoh and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves", *Commentarii Mathematici Universitatis Sancti Pauli*, **47** (1998), 81-92.

[35] I. Semaev, "Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve in characteristic p", *Mathematics of Computation*, **67** (1998), 353-356.

[36] J. Silverman, "The xedni calculus and the elliptic curve discrete logarithm problem", *Designs, Codes and Cryptography*, **20** (2000), 5-40.

[37] R. Silverman and J. Stapleton, Contribution to ANSI X9F1 working group, 1997.

[38] J. Silverman and J. Suzuki, "Elliptic curve discrete logarithms and the index calculus", *Advances in Cryptology – Asiacrypt '98*, Lecture Notes in Computer Science, **1514** (1999), Springer-Verlag, 110-125.

[39] N. Smart, "The discrete logarithm problem on elliptic curves of trace one", *Journal of Cryptology*, **12** (1999), 193-196.

[40] A. Stein, "Equivalences between elliptic curves and real quadratic congruence function fields", *Journal de Théorie des Nombres de Bordeaux*, **9** (1997), 75-95.

[41] A. Stein, V. Müller and C. Thiel, "Computing discrete logarithms in real quadratic congruence function fields of large genus", *Mathematics of Computation*, **68** (1999), 807-822.

[42] E. Teske, "Speeding up Pollard's rho method for computing discrete logarithms", *Algorithmic Number Theory*, Lecture Notes in Computer Science, **1423** (1998), Springer-Verlag, 541-554.

[43] S. Vaudenay, "Hidden collisions on DSS", *Advances in Cryptology – Crypto '96*, Lecture Notes in Computer Science, **1109** (1996), Springer-Verlag, 83-88.

[44] M. Wiener and R. Zuccherato, "Faster attacks on elliptic curve cryptosystems", *Selected Areas in Cryptography*, Lecture Notes in Computer Science, **1556** (1999), Springer-Verlag, 190-200.

[45] R. Zuccherato, "The equivalence between elliptic curve and quadratic function field discrete logarithms in characteristic 2", *Algorithmic Number Theory*, Lecture Notes in Computer Science, **1423** (1998), Springer-Verlag, 621-638.