Certicom ECC Challenge

Abstract

Certicom is pleased to present the Certicom Elliptic Curve Cryptosystem (ECC) Challenge. The first of its kind, the ECC Challenge has been developed to increase the industry's understanding and appreciation for the difficulty of the elliptic curve discrete logarithm problem, and to encourage and stimulate further research in the security analysis of elliptic curve cryptosystems.

It is our hope that the knowledge and experience gained from this Challenge will help confirm comparisons of the security levels of systems such as ECC, RSA and DSA that have been based primarily on theoretical considerations. We also hope it will provide additional information to users of elliptic curve public-key cryptosystems in terms of selecting suitable key lengths for a desired level of security.

The Certicom ECC Challenge Defined

The Challenge is to compute the ECC private keys from the given list of ECC public keys and associated system parameters. This is the type of problem facing an adversary who wishes to completely defeat an elliptic curve cryptosystem.

There are two Challenge Levels: Level I, comprising 109-bit and 131-bit challenges; and Level II, comprising 163-bit, 191-bit, 239-bit and 359-bit challenges. The 109-bit challenges are considered feasible and could be solved within a few months, while the 131-bit challenges will require significantly more resources to solve. All Level II challenges are believed to be computationally infeasible.

The Certicom ECC Challenge is preceded by some Exercises: 79-bit, 89-bit and 97-bit, respectively. These Exercises are feasible to complete given the current state of knowledge in algorithmic number theory and the computational resources available to the industry. Certicom believes that it is feasible that the 79-bit exercises could be solved in a matter of hours, the 89-bit exercises could be solved in a matter of days, and the 97-bit exercises in a matter of weeks using a network of 3000 computers.

Participants can attempt solving the Exercise and Challenge sets using one or both of two finite fields. The first involves elliptic curves over the finite field $\mathbb{F}_{2^m}$ (the field having 2^m elements in it), and the second involves elliptic curves over the finite field $\mathbb{F}_p$ (the field of integers modulo an odd prime p).

The following sections present further background on the Certicom ECC Challenge, a mathematical overview of the elliptic curve discrete logarithm problem, a detailed technical description of the Challenge, the Challenge lists and corresponding prizes, and details on how to report solutions.

Contents

1 Introduction

2 Background

Since the invention of public-key cryptography in 1976 by Whitfield Diffie and Martin Hellman, numerous public-key cryptographic systems have been proposed. All of these systems rely on the difficulty of a mathematical problem for their security.

Over the years, many of the proposed public-key cryptographic systems have been broken, and many others have been demonstrated to be impractical. Today, only three types of systems should be considered both secure and efficient. Examples of such systems, classified according to the mathematical problem on which they are based, are:

1. **Integer factorization problem (IFP):** RSA and Rabin-Williams.

2. **Discrete logarithm problem (DLP):** the U.S. government's Digital Signature Algorithm (DSA), the Diffie-Hellman and MQV key agreement schemes, the ElGamal encryption and signature schemes, and the Schnorr and Nyberg-Rueppel signature schemes.

3. **Elliptic curve discrete logarithm problem (ECDLP):** the elliptic curve analogue of the DSA (ECDSA), and the elliptic curve analogues of the Diffie-Hellman and MQV key agreement schemes, the ElGamal encryption and signature schemes, and the Schnorr and Nyberg-Rueppel signature schemes.

None of these problems have been *proven* to be intractable (i.e., difficult to solve in an efficient manner). Rather, they are *believed* to be intractable because years of intensive study by leading mathematicians and computer scientists around the world has failed to yield efficient algorithms for solving them. As more effort is expended over time in studying and understanding these problems, our confidence in the security of the corresponding cryptographic systems will continue to grow.

3 Elliptic curve cryptosystems

Elliptic curve cryptosystems (ECC) were proposed independently in 1985 by Victor Miller [Miller] and Neal Koblitz [Koblitz]. At the time, both Miller and Koblitz regarded the concept of ECC as mathematically elegant, however felt that its implementation would be impractical. Since 1985, ECC has received intense scrutiny from cryptographers, mathematicians, and computer scientists around the world. On the one hand, the fact that no significant weaknesses have been found has led to high confidence in the security of ECC. On the other hand, great strides have been made in improving the efficiency of the system, to the extent that today ECC is not just practical, but it is the most efficient public-key system known.

The primary reason for the attractiveness of ECC over systems such as RSA and DSA is that the best algorithm known for solving the underlying mathematical problem (namely, the ECDLP) takes *fully exponential* time. In contrast, *subexponential-time* algorithms are known for underlying mathematical problems on which RSA and DSA are based, namely the integer factorization (IFP) and the discrete logarithm (DLP) problems. This means that the algorithms for solving the ECDLP become infeasible much more rapidly as the problem size increases than those algorithms for the IFP and DLP. For this reason, ECC offers security equivalent to RSA and DSA while using far smaller key sizes.

The attractiveness of ECC will increase relative to other public-key cryptosystems as computing power improvements force a general increase in the key size. The benefits of this higher-strength-per-bit include:

- higher speeds,
- lower power consumption,
- bandwidth savings,
- storage efficiencies, and
- smaller certificates.

These advantages are particularly beneficial in applications where bandwidth, processing capacity, power availability, or storage are constrained. Such applications include:

- chip cards,
- electronic commerce,
- web servers,
- cellular telephones, and
- pagers.

4 Why have a challenge?

The objectives of this ECC challenge are the following:

1. To increase the cryptographic community's understanding and appreciation of the difficulty of the ECDLP.
2. To confirm comparisons of the security levels of systems such as ECC, RSA and DSA that have been made based primarily on theoretical considerations.
3. To provide information on how users of elliptic curve public-key cryptosystems should select suitable key lengths for a desired level of security.
4. To determine whether there is any significant difference in the difficulty of the ECDLP for elliptic curves over $\mathbb{F}_{2^m}$ and the ECDLP for elliptic curves over $\mathbb{F}_p$.
5. To determine whether there is any significant difference in the difficulty of the ECDLP for random elliptic curves over $\mathbb{F}_{2^m}$ and the ECDLP for Koblitz curves.
6. To encourage and stimulate research in computational and algorithmic number theory and, in

particular, the study of the ECDLP.

5 The Elliptic Curve Discrete Logarithm Problem (ECDLP)

This section provides a brief overview of the state-of-the-art in algorithms known for solving the elliptic curve discrete logarithm problem. For more information, the reader is referred to Chapter 3 of the *Handbook of Applied Cryptography* [MVV].

6 The discrete logarithm problem

Roughly speaking, the *discrete logarithm problem* is the problem of "inverting" the process of exponentiation. The problem can be posed in a variety of algebraic settings. The most commonly studied versions of this problem are:

1. *The discrete logarithm problem in a finite field (DLP):* Given a finite field $\mathbb{F}_q$ and elements $g, h \in \mathbb{F}_q$, find an integer l such that $g^l = h$ in $\mathbb{F}_q$, provided that such an integer exists.
2. *The elliptic curve discrete logarithm problem (ECDLP):* Given an elliptic curve E defined over a finite field $\mathbb{F}_q$, and two points $P, Q \in E(\mathbb{F}_q)$, find an integer l such that $lP = Q$ in E, provided that such an integer exists.

On the surface, these two problems look quite different. In the first problem, "multiplicative" notation is used: g^l refers to the process of *multiplying* g by itself l times. In the second problem, "additive" notation is used: lP refers to the process of *adding* P to itself l times.

If one casts these notational differences aside, then the two problems are abstractly the same. What is intriguing about the two problems, however, is that the second appears to be much more difficult than the first. The fundamental reason for this is that the algebraic objects in the DLP (*finite fields*) are equipped with two basic operations: addition and multiplication of field elements. In contrast, the algebraic objects in the ECDLP (*elliptic curves over finite felds*) are equipped with only one basic operation: addition of elliptic curve points. The additional structure present in the DLP has led to the discovery of the *index-calculus methods,* which have a *subexponential* running time. Elliptic curves do not possess this additional structure, and for this reason noone has been able to apply the index-calculus methods to the ECDLP (except in very special and well-understood cases). This absence of subexponential-time algorithms for the ECDLP, together with efficient implementation of the elliptic curve arithmetic, is precisely the reason that elliptic curve cryptosystems have proven so attractive for practical use.

7 Algorithms known for the ECDLP

This section briefly overviews the algorithms known for the ECDLP. All of these algorithms take *fully exponential time.*

The notation used is the following:

- q is the order of the underlying finite field.

- $\mathbb{F}_q$ is the underlying finite field of order q.
- E is an elliptic curve defined over $\mathbb{F}_q$.
- $E(\mathbb{F}_q)$ is the set of points on E both of whose coordinates are in $\mathbb{F}_q$, together with the point at infinity.
- P is a point in $E(\mathbb{F}_q)$.
- n is the large prime order of the point P.
- Q is another point in $E(\mathbb{F}_q)$.

The ECDLP is: Given q, E, P, n and Q, find an integer l, $0 \le l \le n - 1$, such that $lP = Q$, provided that such an integer exists.

For the remainder of the discussion, we shall only consider instances of the ECDLP for which the integer l exists.

1. *Naive exhaustive search*.
   In this method, one simply computes successive multiples of P: P, $2P$, $3P$, $4P$, ... until Q is obtained. This method can take up to n steps in the worst case.

2. *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.

3. *Pollard's rho algorithm*.
   This algorithm, due to Pollard [Pollard], 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.

4. *Distributed version of Pollard's rho algorithm*.
   Van Oorschot and Wiener [VW] showed how Pollard's rho algorithm can be parallelized so that when the algorithm is run in parallel on m processors, the expected running time of the algorithm is roughly $\sqrt{\pi n/2}/m$ steps. That is, using m processors results in an m-fold speed-up.

   This distributed version of Pollard's rho algorithm is the fastest general-purpose algorithm known for the ECDLP.

5. *Pohlig-Hellman algorithm*.
   This algorithm, due to Pohlig and Hellman [PH], 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). The elliptic curves in the exercises and challenges posed here are all of this type.

6. *A special class of elliptic curves: supersingular curves*.
Menezes, Okamoto and Vanstone [MOV, Menezes] and Frey and Rück [FR] showed how, under mild assumptions, the ECDLP in an elliptic curve E defined over a finite field F_q can be reduced to the DLP in some extension field F_{q^B} for some $B \geq 1$, where the index-calculus algorithms apply. The reduction algorithm is only practical if B is small — this is not the case for most elliptic curves. To ensure that this 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 $q^B - 1$ for all small B for which the DLP in F_{q^B} is intractable ($1 \leq B \leq 2000/(\log_2 q)$ suffices).

   For the very special class of *supersingular elliptic curves*, it is known that $B \leq 6$. It follows that the reduction algorithm yields a subexponential-time algorithm for the ECDLP in supersingular curves.

7. *Another special class of elliptic curves: anomalous curves*.
Smart [Smart] and Satoh and Araki [SA] independently showed that the ECDLP for the special class of anomalous elliptic curves is easy to solve. An *anomalous elliptic curve* over F_q is an ellipic curve over F_q which has exactly q points. The attack does not extend to any other classes of elliptic curves. Consequently, by verifying that the number of points on an elliptic does not equal the number of elements in the underlying field, one can easily ensure that the Smart-Satoh-Araki attack does not apply to a particular curve.

8 Is there a subexponential-time algorithm for ECDLP?

Whether or not there exists a subexponential-time algorithm for the ECDLP is an important unsettled question, and one of great relevance to the security of ECC. It is extremely unlikely that anyone will ever be able to *prove* that no subexponential-time algorithm exists for the ECDLP. (Analogously, it is extremely unlikely that anyone will ever be able to *prove* that no polynomial-time (efficient) algorithm exists for the integer factorization and discrete logarithm problems.) However, much work has been done on the DLP over the past 20 years, and more specifically on the ECDLP over the past 12 years. No subexponential-time algorithm has been discovered for the ECDLP, confirming the widely-held belief that no such algorithm exists.

A summary of the work done on the ECDLP and further references can be found in the Certicom whitepaper [Certicom].

9 The Challenge Explained

This section gives an overview of some of the mathematics that is relevant to this challenge. The format for the challenge parameters presented in Section 4 is also explained.

For further background on finite fields, consult the books by McEliece [McEliece] and Lidl and Niederreiter [LN]. For further background on elliptic curves, consult the books by Koblitz [Koblitz3] and Menezes [Menezes].

10 Elliptic curves over $\mathbb{F}_{2^m}$ - format and examples

10.1 The finite field $\mathbb{F}_{2^m}$

There are many ways to represent the elements of a finite field with 2^m elements. The particular method used in this challenge is called a *polynomial basis representation.*

Let $f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_2x^2 + f_1x + f_0$ (where $f_i \in \{0, 1\}$ for $i = 0, 1, \ldots, m-1$) be an irreducible polynomial of degree m over $\mathbb{F}_2$. That is, $f(x)$ cannot be factored as a product of two polynomials over $\mathbb{F}_2$, each of degree less than m. The polynomial $f(x)$ *is* called the *reduction polynomial.*

The finite field $\mathbb{F}_{2^m}$ is comprised of all polynomials over $\mathbb{F}_2$ of degree less than m:

$$\mathbb{F}_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0 : a_i \in \{0, 1\}\}.$$

The field element $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0$ is usually denoted by the binary string $(a_{m-1}a_{m-2} \ldots a_1a_0)$ of length m, so that

$$\mathbb{F}_{2^m} = \{(a_{m-1}a_{m-2} \ldots a_1a_0) : a_i \in \{0, 1\}\}.$$

Thus the elements of $\mathbb{F}_{2^m}$ can be represented by the set of all binary strings of length m. The multiplicative identity element (1) is represented by the bit string (00. . .01), while the zero element (additive identity) is represented by the bit string of all 0's.

The following arithmetic operations are defined on the elements of $\mathbb{F}_{2^m}$:

- *Addition*: If $a = (a_{m-1}a_{m-2} \ldots a_1a_0)$ and $b = (b_{m-1}b_{m-2} \ldots b_1b_0)$ are elements of $\mathbb{F}_{2^m}$, then $a + b = c = (c_{m-1}c_{m-2} \ldots c_1c_0)$, where $c_i = (a_i + b_i) \bmod 2$. That is, field addition is performed bitwise.
- *Multiplication*: If $a = (a_{m-1}a_{m-2} \ldots a_1a_0)$ and $b = (b_{m-1}b_{m-2} \ldots b_1b_0)$ are elements of $\mathbb{F}_{2^m}$, then $a \bullet b = r = (r_{m-1}r_{m-2} \ldots r_1r_0)$, where the polynomial $r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \ldots + r_1x + r_0$ is the remainder when the polynomial

  $(a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0) \bullet (b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \ldots + b_1x + b_0)$ is divided by $f(x)$ over $\mathbb{F}_2$.
- *Inversion*: If a is a non-zero element in $\mathbb{F}_{2^m}$, the *inverse* of a, denoted a^{-1}, is the unique element $c \in \mathbb{F}_{2^m}$ for which $a \bullet c = 1$.

Example (*The finite field* $\mathbb{F}_{2^4}$)

Let $f(x) = x^4 + x + 1$ be the reduction polynomial. Then the elements of $\mathbb{F}_{2^4}$ are:

(0000) (1000) (0100) (1100) (0010) (1010) (0110) (1110)

(0001) (1001) (0101) (1101) (0011) (1011) (0111) (1111)

Examples of the arithmetic operations in F_{2^4} are:

- (1101) + (1001) = (0100).
- (1101) • (1001) = (1111).
- $(1101)^{-1}$ = (0100).

10.2 Elliptic curves over F_{2^m}

A (non-supersingular) *elliptic curve* $E(F_{2^m})$ over F_{2^m} defined by the parameters $a, b \in F_{2^m}$, $b \neq 0$, is the set of all solutions (x, y), $x, y \in F_{2^m}$, to the equation

$$y^2 + xy = x^3 + ax^2 + b,$$

together with an extra point O, the *point at infinity*.

The set of points $E(F_{2^m})$ forms a group with the following addition rules:

1. O + O = O
2. (x, y) + O = O + $(x, y) = (x, y)$ for all $(x, y) \in E(F_{2^m})$.
3. $(x, y) + (x, x + y)$ = O for all $(x, y) \in E(F_{2^m})$ (i.e., the negative of the point (x, y) is $-(x, y) = (x, x + y)$).
4. (Rule for adding two distinct points that are not inverses of each other)

   Let $P = (x_1, y_1) \in E(F_{2^m})$ and $Q = (x_2, y_2) \in E(F_{2^m})$ be two points such that $x_1 \neq x_2$. Then $P + Q = (x_3, y_3)$, where

   $$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a,$$

   $$y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \text{ and}$$

   $$\lambda = \frac{y_2 + y_1}{x_2 + x_1}.$$

5. (Rule for doubling a point)

   Let $P = (x_1, y_1) \in E(F_{2^m})$ be a point with $x_1 \neq 0$. (If $x_1 = 0$ then $P = -P$, and so $2P$ = O.) Then $2P = (x_3, y_3)$, where

   $$x_3 = \lambda^2 + \lambda + a,$$

   $$y_3 = x_1^2 + (\lambda + 1)x_3, \text{ and}$$

$$\lambda = x_1 + \frac{y_1}{x_1}.$$

Example (*An elliptic curve over* $\mathbb{F}_{2^4}$)

Consider the finite field $\mathbb{F}_{2^4}$ defined by the reduction polynomial $f(x) = x^4 + x + 1$.
$y^2 + xy = x^3 + (0011)x^2 + (0001)$ is an equation for an elliptic curve E over $\mathbb{F}_{2^4}$. Here $a = (0011)$ and $b = (0001)$. The solutions over $\mathbb{F}_{2^4}$ to this equation are:

(0000,0001)	(0001,1100)	(0001,1101)	(1000,0101)	(1000,1101)
(0110,1000)	(0110,1110)	(1100,0101)	(1100,1001)	(1010,0111)
(1010,1101)	(0111,0010)	(0111,0101)	(1111,0000)	(1111,1111)

$E(\mathbb{F}_{2^4})$ has 16 points, including the point at infinity $\mathcal{O}$. The following are examples of the addition law:

- (1100, 0101) + (1000, 1101) = (0001, 1101).
- 2(1100, 0101) = (0111, 0101).

10.3 Format for challenge parameters (the $\mathbb{F}_{2^m}$ case)

This subsection describes the conventions used for representing the challenge parameters for elliptic curves over $\mathbb{F}_{2^m}$. Two types of elliptic curves over $\mathbb{F}_{2^m}$ are included in the challenge: *random curves* and *Koblitz curves.*

Koblitz curves over $\mathbb{F}_{2^m}$ are special types of elliptic curves E defined over $\mathbb{F}_2$ which have exactly 2 points in $E(\mathbb{F}_2)$. They were first proposed for use in elliptic curve cryptography by Koblitz [Koblitz2]; see also [Solinas].

There have not been any mathematical discoveries to date to suggest that the ECDLP for randomly generated elliptic curves is any easier or harder than the ECDLP for Koblitz curves.

Challenge parameters (random curves)

- m — the order of the finite field is 2^m.
- $f(x)$ — the reduction polynomial which defines the polynomial basis representation of $\mathbb{F}_{2^m}$.
- seedE — the seed that was used to generate the parameters a and b (see Algorithm 1 in Section 3.1.4).
- a, b — the field elements which define the elliptic curve E: $y^2 + xy = x^3 + ax^2 + b$.
- seedP — the seed that was used to generate the point P (see Algorithm 3 in Section 3.1.4).
- x_P, y_P — the x- and y-coordinates of the base point P.
- n — the order of the point P; n is a prime number.
- h — the co-factor h (the number of points in $E(\mathbb{F}_{2^m})$ divided by n).

- seedQ — the seed that was used to generate the point Q (see Algorithm 3 in Section 3.1.4).
- x_Q, y_Q — the x- and y-coordinates of the public key point Q.

Challenge parameters (Koblitz curves)

- m — the order of the finite field is 2^m.
- $f(x)$ — the reduction polynomial which defines the polynomial basis representation of $\mathbb{F}_{2^m}$.
- a, b — the field elements which define the elliptic curve $E: y^2 + xy = x^3 + ax^2 + b$.
- seedP — the seed that was used to generate the point P (see Algorithm 3 in Section 3.1.4).
- x_P, y_P — the x- and y-coordinates of the base point P.
- n — the order of the point P; n is a prime number.
- h — the co-factor h (the number of points in $E(\mathbb{F}_{2^m})$ divided by n).
- seedQ — the seed that was used to generate the point Q (see Algorithm 3 in Section 3.1.4).
- x_Q, y_Q — the x- and y-coordinates of the public key point Q.

Data formats

- *Integers* are represented in hexadecimal, the rightmost bit being the least significant bit. Example: The decimal integer 123456789 is represented in hexadecimal as 075BCD15.
- *Field elements* (of $\mathbb{F}_{2^m}$) are represented in hexadecimal, padded with 0's on the left.
  Example: Suppose $m = 23$. The field element $a = x^{22} + x^{21} + x^{19} + x^{17} + x^5 + 1$ is represented in binary as (11010100000000000100001), or in hexadecimal as 006A0021.
- *Seeds* for generating random elliptic curves and random elliptic curve points (see Section 3.1.4) are 160-bit strings and are represented in hexadecimal.

10.4 Random elliptic curves and points (the $\mathbb{F}_{2^m}$ case)

This subsection describes the method that is used for *verifiably* selecting elliptic curves and points at random. The defining parameters of the elliptic curve or point are defined to be outputs of the one-way hash function SHA-1 (as specified in FIPS 180-1 [SHA-1]). The input seed to SHA-1 then serves as proof (under the assumption that SHA-1 cannot be inverted) that the elliptic curve or point were indeed generated at random.

The following notation is used: $s = \lfloor (m-1)/160 \rfloor$ and $h = m - 160 \bullet s$.

Algorithm 1: Generating a random elliptic curve over $\mathbb{F}_{2^m}$

Input: A field size $q = 2^m$.

Output: A 160-bit bit string seedE and field elements $a, b \in \mathsf{F}_{2m}$ which define an elliptic curve E over F_{2m}.

1. Choose an arbitrary bit string seedE of length 160 bits.
2. Compute H = SHA-1(seedE), and let b_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.
3. For i from 1 to s do:
   Compute b_i = SHA-1((seedE + i) mod 2^{160}).
4. Let b be the field element obtained by the concatenation of $b_0, b_1, \ldots, b_s$ as follows:
   $$b = b_0 \,||\, b_1 \,||\, \ldots \,||\, b_s.$$
5. If $b = 0$ then go to step 1.
6. Let a be an arbitrary element of F_{2m}.
   (Note: For a fixed b, there are only 2 essentially different choices for a — other values of a give rise to *isomorphic* elliptic curves. Hence the choice of a is essentially without loss of generality.)
7. The elliptic curve chosen over F_{2m} is
   $$E : y^2 + xy = x^3 + ax^2 + b.$$
8. Output(seedE, a, b).

Algorithm 2: Verifying that an elliptic curve was randomly generated

Input: A field size $q = 2^m$, a bit string seedE of length 160 bits, and field elements $a, b \in \mathsf{F}_{2m}$ which define an elliptic curve $E : y^2 + xy = x^3 + ax^2 + b$ over F_{2m}.
Output: Acceptance or rejection that E was randomly generated using Algorithm 1.

1. Compute H = SHA-1(seedE), and let b_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.
2. For i from 1 to s do:
   Compute b_i = SHA-1((seedE + i) mod 2^{160}).
3. Let b' be the field element obtained by the concatenation of $b_0, b_1, \ldots, b_s$ as follows:
   $$b' = b_0 \,||\, b_1 \,||\, \ldots \,||\, b_s.$$
4. If $b = b'$ then accept; otherwise reject.

Algorithm 3: Generating a random elliptic curve point

Input: Field elements $a, b \in \mathsf{F}_{2m}$ which define an elliptic curve $E : y^2 + xy = x^3 + ax^2 + b$ over F_{2m}. The order of $E(\mathsf{F}_{2m})$ is $n \bullet h$, where n is a prime.
Output: A bit string seedP, a field element y_U, and a point $P \in E(\mathsf{F}_{2m})$ of order n.

1. Choose an arbitrary bit string seedE of length 160 bits.

2. Compute H = SHA-1(seedP), and let x_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.

3. For i from 1 to s do:
   Compute x_i = SHA-1((seedP + i) mod 2^{160}).

4. Let x_U be the field element obtained by the concatenation of $x_0, x_1, \ldots, x_s$ as follows:
$$x_U = x_0 \| x_1 \| \ldots \| x_s.$$

5. If the equation $y^2 + x_U y = x_U^3 + a x_U^2 + b$ does not have a solution $y \in \mathbb{F}_{2^m}$, then go to step 1.

6. Select an arbitrary solution $y_U \in \mathbb{F}_{2^m}$ to the equation $y^2 + x_U y = x_U^3 + a x_U^2 + b$.
   (Note: this equation will have either 1 or 2 distinct solutions. Hence the choice of y_U is essentially without loss of generality.)

7. Let U be the point (x_U, y_U).

8. Compute $P = hU$.

9. If $P = \mathcal{O}$ then go to step 1.

10. Output(seedP, y_U, P).

Algorithm 4: Verifying that an elliptic curve point was randomly generated

Input: A field size $q = 2^m$, field elements $a, b \in \mathbb{F}_{2^m}$ which define an elliptic curve $E : y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_{2^m}$, a bit string seedP of length 160 bits, a field element $y_U \in \mathbb{F}_{2^m}$, and an elliptic curve point $P = (x_P, y_P)$. The order of $E(\mathbb{F}_{2^m})$ is $n \bullet h$, where n is a prime.
Output: Acceptance or rejection that P was randomly generated using Algorithm 3.

1. Compute H = SHA-1(seedP), and let x_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.

2. For i from 1 to s do:
   Compute x_i = SHA-1((seedP + i) mod 2^{160}).

3. Let x_U be the field element obtained by the concatenation of $x_0, x_1, \ldots, x_s$ as follows:
$$x_U = x_0 \| x_1 \| \ldots \| x_s.$$

4. Let U be the point (x_U, y_U).

5. Verify that U satisfies the equation $y^2 + xy = x^3 + ax^2 + b$.

6. Compute $P' = hU$.

7. If $P \neq P'$ then reject.

8. Accept.

11 Elliptic curves over $\mathbb{F}_p$ - format and examples

11.1 The finite field $\mathbb{F}_p$

Let p be a prime number. The finite field $\mathbb{F}_p$ is comprised of the set of integers

$$\{0, 1, 2, \ldots, p-1\}$$

with the following arithmetic operations:

- *Addition*: If $a, b \in \mathbb{F}_p$, then $a + b = r$, where r is the remainder when $a + b$ is divided by p and $0 \leq r \leq p - 1$. This is known as addition modulo p.
- *Multiplication:* If $a, b \in \mathbb{F}_p$, then $a \bullet b = s$, where s is the remainder when $a \bullet b$ is divided by p and $0 \leq s \leq p - 1$. This is known as multiplication modulo p.
- *Inversion:* If a is a non-zero element in $\mathbb{F}_p$, the *inverse* of a modulo p, denoted a^{-1}, is the unique integer $c \in \mathbb{F}_p$ for which $a \bullet c = 1$.

Example (*The finite field* $\mathbb{F}_{23}$)
The elements of $\mathbb{F}_{23}$ are $\{0, 1, 2, \ldots, 22\}$. Examples of the arithmetic operations in $\mathbb{F}_{23}$ are:

- $12 + 20 = 9$.
- $8 \bullet 9 = 3$.
- $8^{-1} = 3$.

11.2 Elliptic curves over $\mathbb{F}_p$

Let $p > 3$ be a prime number. Let $a, b \in \mathbb{F}_p$ be such that $4a^3 + 27b^2 \neq 0$ in $\mathbb{F}_p$. An *elliptic curve* $E(\mathbb{F}_p)$ over $\mathbb{F}_p$ defined by the parameters a and b is the set of all solutions (x, y), $x, y \in \mathbb{F}_p$, to the equation

$$y^2 = x^3 + ax + b,$$

together with an extra point $\mathcal{O}$, the *point at infinity*.

The set of points $E(\mathbb{F}_p)$ forms a group with the following addition rules:

1. $\mathcal{O} + \mathcal{O} = \mathcal{O}$
2. $(x, y) + \mathcal{O} = \mathcal{O} + (x, y) = (x, y)$ for all $(x, y) \in E(\mathbb{F}_p)$.
3. $(x, y) + (x, -y) = \mathcal{O}$ for all $(x, y) \in E(\mathbb{F}_p)$ (i.e., the negative of the point (x, y) is $-(x, y) = (x, -y)$).
4. (Rule for adding two distinct points that are not inverses of each other)

   Let $P = (x_1, y_1) \in E(\mathbb{F}_p)$ and $Q = (x_2, y_2) \in E(\mathbb{F}_p)$ be two points such that $x_1 \neq x_2$. Then $P + Q = (x_3, y_3)$, where

   $$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \text{ and}$$

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}.$$

5. (Rule for doubling a point)

   Let $P = (x_1, y_1) \in E(\mathsf{F}_p)$ be a point with $y_1 \neq 0$. (If $y_1 = 0$ then $P = -P$, and so $2P = \mathsf{O}$.) Then $2P = (x_3, y_3)$, where

$$x_3 = \lambda^2 - 2x_1,$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \text{ and}$$

$$\lambda = \frac{3x_1^2 + a}{2y_1}.$$

Example (*An elliptic curve over* F_{23})
$y^2 = x^3 + x + 1$ is an equation for an elliptic curve E over F_{23}. Here $a = 1$ and $b = 1$. The solutions over F_{23} to this equation are:

(0, 1)	(0, 22)	(1, 7)	(1, 16)	(3, 10)	(3, 13)	(4, 0)	(5, 4)	(5, 19)
(6, 4)	(6, 19)	(7, 11)	(7, 12)	(9, 7)	(9, 16)	(11, 3)	(11, 20)	(12, 4)
(12, 19)	(13, 7)	(13, 16)	(17, 3)	(17, 20)	(18, 3)	(18, 20)	(19, 5)	(19, 18)

$E(\mathsf{F}_{23})$ has 28 points, including the point at infinity O. The following are examples of the addition law:

- (3, 10) + (9, 7) = (17, 20).
- 2(3, 10) = (7, 12).

11.3 Format for challenge parameters (the F_p case)

This subsection describes the conventions used for representing the challenge parameters for elliptic curves over F_p.

Challenge parameters

- p — the order of the finite field; p is a prime number.
- seedE — the seed that was used to generate the parameters a and b (see Algorithm 5 in Section 3.2.4).
- a, b — the field elements which define the elliptic curve E: $y^2 = x^3 + ax + b$.
- seedP — the seed that was used to generate the point P (see Algorithm 7 in Section 3.2.4).
- x_P, y_P — the x- and y-coordinates of the base point P.

- n — the order of the point P; n is a prime number.
- h — the co-factor h (the number of points in $E(\mathbb{F}_p)$ divided by n).
- seedQ — the seed that was used to generate the point Q (see Algorithm 7 in Section 3.2.4).
- x_Q, y_Q — the x- and y-coordinates of the public key point Q.

Data formats

- *Integers* are represented in hexadecimal, the rightmost bit being the least significant bit. Example: The decimal integer 123456789 is represented in hexadecimal as 075BCD15.
- *Field elements* (of $\mathbb{F}_p$) are represented as hexadecimal integers.
- *Seeds* for generating random elliptic curves and random elliptic curve points (see Section 3.2.4) are 160-bit strings and are represented in hexadecimal.

11.4 Random elliptic curves and points (the $\mathbb{F}_p$ case)

This subsection describes the method that is used for *verifiably* selecting elliptic curves and points at random. The defining parameters of the elliptic curve or point are defined to be outputs of the one-way hash function SHA-1 (as specified in FIPS 180-1 [SHA-1]). The input seed to SHA-1 then serves as proof (under the assumption that SHA-1 cannot be inverted) that the elliptic curve or point were indeed generated at random.

The following notation is used: $t = \lceil \log_2 p \rceil$, $s = \lfloor (t-1)/160 \rfloor$ and $h = t - 160 \bullet s$.

Algorithm 5: Generating a random elliptic curve over $\mathbb{F}_p$

Input: A field size p, where p is a prime.
Output: A 160-bit bit string seedE and field elements $a, b \in \mathbb{F}_p$ which define an elliptic curve E over $\mathbb{F}_p$.

1. Choose an arbitrary bit string seedE of length 160 bits.
2. Compute H = SHA-1(seedE), and let c_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.
3. Let W_0 denote the bit string of length h bits obtained by setting the leftmost bit of c_0 to 0. (This ensures that $r < p$.)
4. For i from 1 to s do:
   Compute W_i = SHA-1((seedE + i) mod 2^{160}).
5. Let W be the bit string obtained by the concatenation of $W_0, W_1, \ldots, W_s$ as follows:
   $$W = W_0 \,\|\, W_1 \,\|\, \ldots \,\|\, W_s.$$
6. Let $w_1, w_2, \ldots, w_t$ be the bits of W from leftmost to rightmost. Let r be the integer
   $r = \sum_{i=1}^{t} w_i 2^{t-i}$.

7. Choose arbitrary integers $a, b \in \mathbb{F}_p$ such that $r \bullet b^2 \equiv a^3 \pmod{p}$.

   (Note: For a fixed $r \neq 0$, there are only 2 essentially different choices for a and b — other values of a and b give rise to *isomorphic* elliptic curves. Hence the choice of a and b is essentially without loss of generality.)

8. If $4a^3 + 27\,b^2 \equiv 0 \pmod{p}$ then go to step 1.

9. The elliptic curve chosen over $\mathbb{F}_p$ is

$$E : y^2 = x^3 + ax + b.$$

10. Output(seedE, a, b).

Algorithm 6: Verifying that an elliptic curve was randomly generated

Input: A field size p (a prime), a bit string seedE of length 160 bits, and field elements $a, b \in \mathbb{F}_p$ which define an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$.
Output: Acceptance or rejection that E was randomly generated using Algorithm 5.

1. Compute H = SHA-1(seedE), and let c_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.
2. Let W_0 denote the bit string of length h bits obtained by setting the leftmost bit of c_0 to 0.
3. For i from 1 to s do:
   Compute W_i = SHA-1((seedE + i) mod 2^{160}).
4. Let W' be the bit string obtained by the concatenation of $W_0, W_1, \ldots, W_s$ as follows:
   $$W' = W_0 \| W_1 \| \ldots \| W_s.$$
5. Let $w_1, w_2, \ldots, w_t$ be the bits of W from leftmost to rightmost. Let r' be the integer $r' = \sum_{i=1}^{t} w_i 2^{t-i}$.
6. If $r' \bullet b^2 \equiv a^3 \pmod{p}$ then accept; otherwise reject.

Algorithm 7: Generating a random elliptic curve point

Input: Field elements $a, b \in \mathbb{F}_p$ which define an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$. The order of $E(\mathbb{F}_p)$ is $n \bullet h$, where n is a prime.
Output: A bit string seedP, a field element y_U, and a point $P \in E(\mathbb{F}_p)$ of order n.

1. Choose an arbitrary bit string seedP of length 160 bits.
2. Compute H = SHA-1(seedP), and let c_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.
3. Let x_0 denote the bit string of length h bits obtained by setting the leftmost bit of c_0 to 0.
4. For i from 1 to s do:
   Compute x_i = SHA-1((seedE + i) mod 2^{160}).
5. Let x_U be the bit string obtained by the concatenation of $x_0, x_1, \ldots, x_s$ as follows:
   $$x_U = x_0 \| x_1 \| \ldots \| x_s.$$
6. If the equation $y^2 = x_U^3 + ax_U + b$ does not have a solution $y \in \mathbb{F}_p$, then go to step 1.
7. Select an arbitrary solution $y_U \in \mathbb{F}_p$ to the equation $y^2 = x_U^3 + ax_U + b$.
   (Note: this equation will have either 1 or 2 distinct solutions. Hence the choice of y_U is essentially without loss of generality.)

8. Let U be the point (x_U, y_U).

9. Compute $P = hU$.

10. If $P = \mathcal{O}$ then go to step 1.

11. Output(seedP, y_U, P).

Algorithm 8: Verifying that an elliptic curve point was randomly generated

Input: A field size p (a prime), field elements $a, b \in \mathbb{F}_p$ which define an elliptic curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, a bit string seedP of length 160 bits, a field element $y_U \in \mathbb{F}_p$, and an elliptic curve point $P = (x_P, y_P)$. The order of $E(\mathbb{F}_p)$ is $n \bullet h$, where n is a prime.
Output: Acceptance or rejection that P was randomly generated using Algorithm 7.

1. Compute H = SHA-1(seedP), and let c_0 denote the bit string of length h bits obtained by taking the h rightmost bits of H.

2. Let x_0 denote the bit string of length h bits obtained by setting the leftmost bit of c_0 to 0.

3. For i from 1 to s do:
   Compute x_i = SHA-1((seedE + i) mod 2^{160}).

4. Let x_U be the bit string obtained by the concatenation of $x_0, x_1, \ldots, x_s$ as follows:
$$x_U = x_0 \| x_1 \| \ldots \| x_s.$$

5. Let U be the point (x_U, y_U).

6. Verify that U satisfies the equation $y^2 = x^3 + ax + b$.

7. Compute $P' = hU$.

8. If $P \neq P'$ then reject.

9. Accept.

12 Further details about the challenge

This subsection presents some more information about the challenge. Each problem posed is to compute the private key given the elliptic curve parameters, the base point P of order n, and the public key point Q. The *private key* is the *unique* integer l, $0 \le l \le n - 1$, such that $Q = lP$. Each problem is therefore an instance of the elliptic curve discrete logarithm problem (ECDLP); see Section 2.

With the exception of the Koblitz curves, all elliptic curves have been chosen randomly in a *verifiable* manner (see Sections 3.1.4 and 3.2.4) — anyone can verify that the elliptic curve parameters were indeed generated at random.

Another interesting feature of the challenge is that the points P and Q having order n were also chosen randomly in a *verifiable* manner (see Sections 3.1.4 and 3.2.4). This means that each particular private key l is presently unknown even to the creators of the challenge!! However, any alleged solution

l' that is found to a challenge can easily be verified by checking that $Q = l'P$. The challenges presented here therefore adhere to the philosophy expressed by Matt Blaze [Blaze] at Crypto '97 that the solutions to a challenge should be unknown to the creators at the outset of the challenge.

The problems have been separated into two categories:

i) elliptic curves over $\mathbb{F}_{2^m}$, and

ii) elliptic curves over $\mathbb{F}_p$.

There have not been any mathematical discoveries to date to suggest that the ECDLP for elliptic curves over $\mathbb{F}_{2^m}$ is any easier or harder that the ECDLP for elliptic curves over $\mathbb{F}_p$.

For each of these categories, the problems have been further divided into three sub-categories:

i) Exercises,

ii) Level I Challenges, and

iii) Level II Challenges.

These are distinguished by the size of the parameter n, the prime order of the base point P. As the size of n increases, the problem is expected to become harder. By a *k-bit challenge,* we shall mean a challenge whose parameter n has bitlength k.

13 Time estimates for exercises and challenges

This subsection provides a *very rough* estimate for the time to solve a k-bit challenge with parameter n.

Recall from Section 2.2 that the distributed version of Pollard’s rho algorithm using m processors takes approximately $\sqrt{\pi n/2}/m$ steps. Here, each “step” is an elliptic curve addition or double. Thus, if a computer can perform a point operation in s microseconds, then the number of computer days required before a discrete logarithm is found is expected to be roughly

$$\frac{s \times 10^{-6}}{60 \times 60 \times 24} \times \sqrt{\pi n/2} \approx s \times 2^{-36} \times \sqrt{n} \text{ machine days.}$$

To illustrate this, consider solving an instance of the ECDLP over $\mathbb{F}_{2^{89}}$ with $n \approx 2^{89}$. A fast implementation of elliptic curve operations on a widely available computer, say a Pentium 100, for a curve over $\mathbb{F}_{2^{89}}$ might take on the order of 50 micro seconds for a point operation. Thus such an implementation would require

$$50 \times 2^{-36} \times \sqrt{2^{89}} \approx 18100 \text{ machine days}$$

to find a single discrete logarithm. So, for example, one such machine running 24 hours a day would require 18100 days. A network of 3000 such machines would require about 6 days.

An implementation report of the Pollard rho algorithm for solving the ECDLP can be found in [HMV].

14 Exercise Lists and Challenge Lists

15 Elliptic curves over $\mathbb{F}_{2^m}$

In the following tables, ECC2-*k* denotes that the exercise or challenge is over a field $\mathbb{F}_{2^m}$, and that the parameter *n* has bitlength *k*. Furthermore, ECC2K-*k* denotes that the elliptic curve used is a Koblitz curve (see Section 3.1.3), rather than a randomly generated curve.

For a description of the format of the challenge parameters, see Section 3.1.3. For further details about the challenge, see Section 3.3. The time estimates for each exercise and challenge were derived as in Section 3.4.

Using these timings, it is expected that the 79-bit exercise could be solved in a matter of hours, the 89-bit exercise could be solved in a matter of days, and the 97-bit exercise in a matter of weeks using a network of 3000 computers.

The 109-bit Level I challenge is feasible using a very large network of computers. The 131-bit Level I challenge is expected to be infeasible against realistic software and hardware attacks, unless of course a new algorithm for the ECDLP is discovered.

The Level II challenges are infeasible given today's computer technology and knowledge. The elliptic curves for these challenges meet the stringent security requirements imposed by forthcoming ANSI banking standards [X962, X963].

15.1 Exercises

Exercise	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2-79	79	565	*Handbook of Applied Cryptography & Maple V software*
ECC2-89	89	18100	*Handbook of Applied Cryptography & Maple V software*
ECC2K-95	97	144815	$ 5,000
ECC2-97	97	289000	$ 5,000

15.2 Level I challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2K-108	109	1.31×10^7	$ 10,000
ECC2-109	109	1.85×10^7	$ 10,000

ECC2K-130	131	2.68×10^{10}	$ 20,000
ECC2-131	131	3.79×10^{10}	$ 20,000

15.3 Level II challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2-163	163	2.48×10^{15}	$ 30,000
ECC2K-163	163	2.48×10^{15}	$ 30,000
ECC2-191	191	4.07×10^{19}	$ 40,000
ECC2-238	239	4.84×10^{26}	$ 50,000
ECC2K-238	239	4.84×10^{26}	$ 50,000
ECC2-353	359	9.86×10^{43}	$ 100,000
ECC2K-358	359	5.58×10^{44}	$ 100,000

16 Elliptic curves over $\mathbb{F}_p$

In the following tables, ECCp-*k* denotes that the exercise or challenge is over a field $\mathbb{F}_p$ (*p* prime), and that the parameter *n* has bitlength *k*.

For a description of the format of the challenge parameters, see Section 3.2.3. For further details about the challenge, see Section 3.3. The time estimates for each exercise and challenge were derived as in Section 3.4.

Using these timings, it is expected that the 79-bit exercise could be solved in a matter of hours, the 89-bit exercise could be solved in a matter of days, and the 97-bit exercise in a matter of weeks using a network of 3000 computers.

The 109-bit Level I challenge is feasible using a very large network of computers. The 131-bit Level I challenge is expected to be infeasible against realistic software and hardware attacks, unless of course a new algorithm for the ECDLP is discovered.

The Level II challenges are infeasible given today's computer technology and knowledge. The elliptic curves for these challenges meet the stringent security requirements imposed by forthcoming ANSI banking standards [X962, X963].

16.1 Exercises

Exercise	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-79	79	565	*Handbook of Applied Cryptography & Maple V software*

ECCp-89	89	18100	*Handbook of Applied Cryptography & Maple V software*
ECCp-97	97	289000	$ 5,000

16.2 Level I challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-109	109	1.85×10^{7}	$ 10,000
ECCp-131	131	3.79×10^{10}	$ 20,000

16.3 Level II challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-163	163	2.48×10^{15}	$ 30,000
ECCp-191	191	4.07×10^{19}	$ 40,000
ECCp-239	239	6.83×10^{26}	$ 50,000
ECCp-359	359	7.88×10^{44}	$ 100,000

17 Challenge Rules

18 The Rules and Reporting a Solution

Each exercise and challenge in the Exercise and Challenge Lists is based on the problem of computing the ECC private key from the given ECC public key and associated system parameters. An individual or group of individuals reporting a solution must also provide a full explanation of how that solution was reached. No reported solutions will be accepted without a detailed explanation of the steps taken and calculations made to find an ECC private key.

As noted in Section 3.3, each particular private key is presently unknown even to the creators of the Certicom ECC Challenge. Unique to all algorithms based on the discrete logarithm problem, a supposed ECC public key can be validated to ensure it conforms to the arithmetic requirements of a public-key. This validation is 100%. When an ECC public key is validated, it is known that a private key for the public key can logically exist. This capability of key validation is used in the Certicom ECC Challenge.

The proposed solution must be sent via email to Certicom Corp., following the Format of Submissions specified in Section 5.1.1. The correct solution for an Exercise or Challenge will be the one that was received first by Certicom Corp. and checked by an independent, third-party appointed by Certicom.

Certicom Corp. reserves the right to change the contest rules at any time at its sole discretion, without notice, including the right to change or extend the challenge lists, to change the prize amounts, and/or to terminate the contest. While Certicom has appointed an independent, third-party to check the solutions, Certicom Corp. is the sole arbiter and administrator for this contest. Certicom's judgement in all matters is final.

Queries on the Certicom ECC Challenge can be addressed to:

Certicom ECC Challenge Administrator
Certicom Corp.
200 Matheson Blvd. West
Mississauga, Ontario
Canada L5R 3L7

For further information concerning the Certicom ECC Challenge, email inquiries can be sent to certicom-ecc-challenge@certicom.com. For news of the latest developments in the Certicom ECC Challenge, check Certicom's web site at www.certicom.com.

18.1 Format of Submissions

All solution submissions for any of the exercises or challenges must be sent by email to certicom-ecc-challenge@certicom.com. The report of a solution should clearly state that the submission is being made for the Certicom ECC Challenge. The body of the email message must contain the following information, titled with the respective headers:

- **Name**: name(s) of the person or people making the submission;
- **Address**: mailing address of the reporting party;
- **Email**: email address of the reporting party;
- **Phone**: telephone number and area code of the reporting party;
- **Exercise or Challenge**: specific exercise or challenge for which the submission is being made (see Sections 4.1 and 4.2 for exercise and challenge tables);
- **Solution**: actual private key value being submitted;
- **Method**: steps and computations taken to calculate the private key, and any other relevant information such as the estimated time taken to calculate the solution and the type of machine(s) used in the computations.

After each field, there must be the word "DONE" to indicate the end of the submission. The "name", "address", "email", "phone", "exercise or challenge", "solution", and "method" fields must be present in every submission. Without these fields, the solution report will be rejected.

While it is preferred that the information fields be separated as specified above, information from two fields can be merged into one. Each field must start on a new line. If more than one person is reporting a solution in a group, the names of each individual along with their corresponding address, email and phone number should be contained in separate fields in alphabetical order.

19 Prizes and Status

The following are the official prize lists for the Certicom ECC Challenge:

19.1 Exercise Prize Lists

Exercise	Field Size (in bits)	Prize (US$)	Start Date	End Date	Time for Solution
ECC2-79	79	*Handbook of Applied Cryptography* and one complete *Maple V software* package for any platform requested	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-79	79	*Handbook of Applied Cryptography* and one complete *Maple V software* package for any platform requested	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-89	89	*Handbook of Applied Cryptography* and one complete *Maple V software* package for any platform requested	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-89	89	*Handbook of Applied Cryptography* and one complete *Maple V software* package for any platform requested	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-97	97	$5,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-95	97	$5,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-97	97	$5,000	Thursday, November 6, 1997, 1 p.m. EST		

Note: the *Handbook of Applied Cryptography*, co-authored by Dr. Alfred J. Menezes, Dr. Paul C. van Oorshcot, and Dr. Scott A. Vanstone, has a retail value of US$75 per copy. The Maple V software is a leading cryptographic research tool and has a retail value of $1000 and $2000 US, depending upon the platform.

19.2 Level I Challenge Prize List

Challenge	Field Size (in bits)	Prize (US$)	Start Date	End Date	Time for Solution
ECC2-109	109	$10,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-108	109	$10,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-109	109	$10,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-131	131	$20,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-130	131	$20,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-131	131	$20,000	Thursday, November 6, 1997, 1 p.m. EST		

19.3 Level II Challenge Prize List

Challenge	Field Size (in bits)	Prize (US$)	Start Date	End Date	Time for Solution
ECC2-163	163	$30,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-163	163	$30,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-163	163	$30,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-191	191	$40,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-191	191	$40,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-238	239	$50,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-238	239	$50,000	Thursday, November 6, 1997, 1 p.m. EST		
ECCp-239	239	$50,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2-353	359	$100,000	Thursday, November 6, 1997, 1 p.m. EST		
ECC2K-358	359	$100,000	Thursday, November 6, 1997, 1 p.m. EST		

19.4 Administration and Collection of Prizes

The first person or party to report the correct solution for any exercise or challenge, complete with the methodology and steps used to discover that solution, will win the prize for that particular exercise or challenge he/she has solved.

An organized group of individuals reporting a solution will be treated the same as one person reporting a solution, in that only one cash prize will be awarded to the group with the correct solution, reported as specified in section 5.1.1. The prize shall be administered so that it is divided evenly among all members of that group.

In several instances, there are two exercises or challenges with the same field size (e.g. 97-bit exercise) and the same corresponding cash prize, but are based on one of two finite fields—elliptic curves over the finite field F_{2^m} and elliptic curves over the finite field F_p. These exercises and challenges have different solutions and the corresponding prizes will be awarded accordingly. Therefore, should the correct solution be properly reported for the Exercise ECC2-97 (97-bit field size over the field F_{2^m}), the ECC2K-97 exercise (97-bit field size over the field F_p) would still be available to solve and the cash prize available for award to the person(s) with the correct solution.

20 References

[Blaze] M. Blaze, "A better DES challenge", presentation at the rump session at Crypto '97.

[Certicom] Certicom Corp. white paper, "Remarks on the security of the elliptic curve cryptosystem", September 1997. Available from http://www.certicom.com

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

[HMV] G. Harper, A. Menezes and S. Vanstone, "Public-key cryptosystems with very small key lengths", *Advances in Cryptology - EUROCRYPT '92*, Lecture Notes in Computer Science, volume 658, Springer-Verlag, pages 163-173, 1993.

[Koblitz] N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, volume 48, pages 203-209, 1987.

[Koblitz2] N. Koblitz, "CM-curves with good cryptographic properties", *Advances in Cryptology -CRYPTO '91*, Lecture Notes in Computer Science, volume 576, Springer-Verlag, pages 279-287, 1992.

[Koblitz3] N. Koblitz, *A Course in Number Theory and Cryptography,* Springer-Verlag, 2nd edition, 1994.

[LN] R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications,* Cambridge University Press, 1994.

[McEliece] R. McEliece, *Finite Fields for Computer Scientists and Engineers,* Kluwer Academic Publishers, 1987.

[Menezes] A. Menezes, *Elliptic Curve Pub1ic Key Cryptosystems,* Kluwer Academic Publishers, 1993.

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

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

[Miller] V. Miller, "Uses of elliptic curves in cryptography", *Advances in Cryptology - CRYPTO '85,* Lecture Notes in Computer Science, volume 218, Springer-Verlag, pages 417-426, 1986.

[VW] P. van Oorschot and M. Wiener, "Parallel collision search with cryptanalytic applications", to appear in *Journal of Cryptology.* (An earlier version appeared in the Proceedings of the 2nd ACM Conference on Computer and Communications Security, ACM Press, pages 210-218, 1994.)

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

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

[SHA-1] FIPS 180-1, "Secure hash standard", Federal Information Processing Standards Publication 180-1, U.S. Department of Commerce/N.I.S.T., April 1995.

[SA] T. Satoh and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves", preprint, 1997.

[Smart] N. Smart, Announcement of an attack on the ECDLP for anomalous elliptic curves, 1997.

[Solinas], J. Solinas, "An improved algorithm for arithmetic on a family of elliptic curves", *Advances in Cryptology - CRYPTO '97,* Lecture Notes in Computer Science, volume 1294, Springer-Verlag, pages 357-371, 1997.

[X962] ANSI X9.62, "The elliptic curve digital signature algorithm (ECDSA)", draft standard, 1997.

[X963] ANSI X9.63, "Elliptic curve key agreement and transport protocols", draft standard, 1997.