Certicom ECC Challenge

Certicom Research

Original date: November 6, 1997
Latest update: November 10, 2009*

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 were all solved by 2004, while the 131-bit challenges will require significantly more resources, though may be soon within reach.[1] All Level II challenges are believed to be computationally infeasible.[2]

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.

*Updates since 2003 include reporting of two solved challenges, typesetting adjustment, minor editorial changes, clarifying footnotes, updated progress on the ECDLP, updated references, and an added appendix with curve definitions.

[1]Formerly, this sentence was "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."

[2]NIST has recommended that 80-bit security be discontinued in 2010, presumably for security reasons, suggesting that NIST deems the 163-bit challenges could be solvable by an adversary.

Contents

1 Introduction

1.1 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[3], 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.

1.2 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

[3]Assertion from 1997.

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.

1.3 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.

2 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].

2.1 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 fields*) 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 no one 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.

2.2 Algorithms known for the ECDLP

This section gives a brief overview of 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, $0 \leq l \leq n-1$, 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, $2P$, $3P$, $4P, \ldots$ 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.

   Gallant, Lambert and Vanstone [GLV], and Wiener and Zuccherato [WZ] 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.

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 $\sqrt{\pi n}/(2m)$ 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. *Pollard's lambda method.*
   This is another randomized algorithm due to Pollard [Pollard]. 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 [VW]. 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 $b < 0.39n$ [VW].

7. *Multiple Logarithms*
   R. Silverman and Stapleton [SS] observed that if a single instance of the ECDLP (for a given elliptic curve E and a base point P) is solved using Pollard's rho method, then the work done in solving this instance can be used to speed up the solution of other instance of the ECDLP for the same curve E and base point P. More precisely, 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, however, 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. *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 $\mathbb{F}_q$ can be reduced to the DLP in some extension field $\mathbb{F}_{q^B}$ for some $B \geq 1$, where the number field sieve algorithm applies. The reduction algorithm is only practical if B is small — this is not the case for most elliptic curves, as shown by Balasubramanian and Koblitz [BK]. 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 $\mathbb{F}_{q^B}$ is tractable ($1 \leq B \leq 2000/(\log_2 q)$ 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. For this reason, supersingular curves are explicitly excluded from use in the ECDSA.

9. *Another special class of elliptic curves: anomalous curves.*
   Semaev [Semaev], Smart [Smart], Satoh and Araki [SA] showed that the ECDLP for the special class of anomalous elliptic curves is easy to solve. An anomalous elliptic curve over $\mathbb{F}_q$ is an elliptic curve over $\mathbb{F}_q$ 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.

10. *Curves defined over a small field.*
    Suppose that E is an elliptic curve defined over the finite field $\mathbb{F}_{2^B}$. Gallant, Lambert and Vanstone [GLV], and Wiener and Zuccherato [WZ] showed how Pollard's rho algorithm for computing elliptic curve logarithms in $E(\mathbb{F}_{2^{Bd}})$ 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, 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 [GS], expanding on earlier work by Frey [Frey], 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. More recently, Gaudry, Hess and Smart [GHS] refined these ideas to provide strong evidence that when m has a small divisor l (say, $l = 4$), the ECDLP for elliptic curves defined over $\mathbb{F}_{2^m}$ can be solved faster than with Pollard's rho algorithm.

    Menezes and Qu [MQ] showed that the GHS attack is ineffective (in the sense of being slower than Pollard's rho algorithm) for for all elliptic curves defined over finite fields $\mathbb{F}_{2^m}$ where m is prime and $m \in [160, 600]$. The GHS attack for elliptic curves over $\mathbb{F}_{2^m}$ where m is composite was extensively analyzed (see [JMS], [MMT] and [MTW]), with the conclusion that the attack is indeed effective in special cases. Hess [Hess] later generalized the GHS attack, leading to additional concerns for certain composite m [MT]. In view of these Weil descent attacks, it seems prudent to avoid use of elliptic curves over finite fields $\mathbb{F}_{2^m}$ where m is composite.

    It should be noted that some ECC standards, including [X962, X963], explicitly exclude the use of elliptic curves over composite fields.

12. *Gaudry's subfield-base index-calculus algorithm.*
Gaudry [Gaudry] proposed an index-calculus attack on the ECDLP for elliptic curves defined over fields F_{q^m} with $m > 1$. In Gaudry's attack, the factor base consists of points whose x-coordinate lies in F_q. For fixed $m \geq 3$, Gaudry's attack has running time $O(q^{2-2/m})$ (see also [Diem1]) which, although not subexponential, is faster than Pollard's ρ method. More recently, Diem [Diem2] proved that Gaudry's algorithm has subexponential running time when the field order q^m increases in such a way that m^2 is of order $\log q$. Gaudry's attack and its derivatives do not seem effective for solving the ECDLP for elliptic curves that are used in practice, such as the NIST elliptic curves.

2.3 Is there a subexponential-time algorithm for ECDLP?

Whether or not there exists a subexponential-time algorithm for the general 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. (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 24 years, and more specifically on the ECDLP over the past 16 years. No subexponential-time algorithm has been discovered for the general ECDLP, confirming the widely-held belief that no such algorithm exists.

In particular, Miller [Miller] and J. Silverman and Suzuki [SS2] 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.

Another very interesting line of attack on the ECDLP, called the *xedni-calculus attack* was recently proposed by J. Silverman [Silverman]. However, it was subsequently shown by a team of researchers including J. Silverman (see Jacobson et al. [JKSST]) that the attack is virtually certain to fail in practice.

For elliptic curves of various special forms, subexponential-time algorithms are known, as outlined in Section 2.2.

3 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].

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

3.1.1 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 $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0$ (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:

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

The field element $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0$ 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 $(00\ldots01)$, 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 $a \cdot b = r = (r_{m-1}r_{m-2}\ldots r_1r_0)$, where the polynomial $r_{m-1}x^{m-1} + r_{m-2}x^{m-2} + \cdots + r_1x + r_0$ is the remainder when the polynomial

  $$(a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0) \cdot (b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0)$$

  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 $a \cdot c = 1$.

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 $\mathbb{F}_{2^4}$ are:

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

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

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

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

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

The set of points $E(\mathbb{F}_{2^m})$ 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 $(x, y) \in E(\mathbb{F}_{2^m})$.
3. $(x, y) + (x, x+y) = \mathcal{O}$ for all $(x, y) \in E(\mathbb{F}_{2^m})$ (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 $P = (x_1, y_1) \in E(\mathbb{F}_{2^m})$ and $Q = (x_2, y_2) \in E(\mathbb{F}_{2^m})$ be two points such that $x_1 \neq x_2$. Then $P + Q = (x_3, y_3)$, where

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a, \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1, \text{ and} \\ \lambda &= \frac{y_2 + y_1}{x_2 + x_1}. \end{aligned}$$

5. (Rule for doubling a point)

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

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + a \\ y_3 &= x_1^2 + (\lambda + 1)x_3, \text{ and} \\ \lambda &= x_1 + \frac{y_1}{x_1}. \end{aligned}$$

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:

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

3.1.3 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].

Apart from the $\sqrt{m}$-fold speedup that can be obtained with the parallelized Pollard's rho method, 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.

3.1.4 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 $h = m - 160 \cdot s$.

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 $a, b \in \mathbb{F}_{2^m}$ which define an elliptic curve E over $\mathbb{F}_{2^m}$.

1. Choose an arbitrary bit string seedE of length 160 bits.
2. Compute $H = \text{SHA-1}(\text{seedE})$, 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 $b_i = \text{SHA-1}((\text{seedE} + i) \bmod 2^{160})$.

4. Let b be the field element obtained by the concatenation of $b_0, b_1, \ldots, b_s$ as follows:

$$b = b_0 \parallel b_1 \parallel \cdots \parallel b_s.$$

5. If $b = 0$ then go to step 1.

6. Let a be an arbitrary element of $\mathbb{F}_{2^m}$.
   (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 $\mathbb{F}_{2^m}$ 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 $a, b \in \mathbb{F}_{2^m}$ which define an elliptic curve $E : y^2 + xy = x^3 + ax^2 + b$ over $\mathbb{F}_{2^m}$.
Output: Acceptance or rejection that E was randomly generated using Algorithm 1.

1. Compute $H = \text{SHA-1(seedE)}$, 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 $b_i = \text{SHA-1}((\text{seedE} + i) \bmod 2^{160})$.

3. Let b' be the field element obtained by the concatenation of $b_0, b_1, \ldots, b_s$ as follows:

$$b' = b_0 \parallel b_1 \parallel \cdots \parallel b_s.$$

4. If $b = b'$ then accept; otherwise reject.

Algorithm 3: Generating a random elliptic curve point

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

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

2. Compute $H = \text{SHA-1(seedP)}$, 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 $x_i = \text{SHA-1}((\text{seedP} + i) \bmod 2^{160})$.

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

$$x_U = x_0 \,\|\, x_1 \,\|\, \cdots \,\|\, x_s.$$

5. If the equation $y^2 + x_U y = x_U^3 + ax_U^2 + b$ 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 $y^2 + x_U y = x_U^3 + ax_U^2 + b$.
   (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. 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 $n \cdot h$, where n is a prime.
Output: Acceptance or rejection that P was randomly generated using Algorithm 3.

1. Compute $H = \text{SHA-1}(\text{seedP})$, 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 $x_i = \text{SHA-1}((\text{seedP} + i) \bmod 2^{160})$.

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

$$x_U = x_0 \,\|\, x_1 \,\|\, \cdots \,\|\, x_s.$$

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 $P \neq P'$ or if $nP \neq \mathcal{O}$ then reject.

8. Accept.

3.2 Elliptic curves over $\mathbb{F}_p$ – format and examples

3.2.1 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 $a \cdot b = s$, where s is the remainder when $a \cdot b$ 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 $a \cdot c = 1$.

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:

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

3.2.2 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

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \text{ and} \\ \lambda &= \frac{y_2 - y_1}{x_2 - x_1}. \end{aligned}$$

5. (Rule for doubling a point)

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

$$\begin{aligned} x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \text{ and} \\ \lambda &= \frac{3x_1^2 + a}{2y_1}. \end{aligned}$$

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

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

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

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

3.2.3 Format for challenge parameters (the $\mathbb{F}_p$ case)

This subsection describes the conventions used for representing the challenge parameters for elliptic curves over $\mathbb{F}_p$.

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.

3.2.4 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 $h = t - 160 \cdot s$.

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

Input: A field size p, where p is 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 $H = \text{SHA-1}(\text{seedE})$, 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 $W_i = \text{SHA-1}((\text{seedE} + i) \bmod 2^{160})$.
5. Let W be the bit string obtained by the concatenation of $W_0, W_1, \ldots, W_s$ as follows:
$$W = W_0 \parallel W_1 \parallel \cdots \parallel W_s.$$
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 $r \cdot b^2 \equiv a^3 \bmod p$.
   (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 $4a^3 + 27b^2 \equiv 0 \pmod{p}$ 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$((\text{seedE} + i) \bmod 2^{160})$.
4. Let W' be the bit string obtained by the concatenation of $W_0, W_1, \ldots, W_s$ as follows:
$$W' = W_0 \,\|\, W_1 \,\|\, \cdots \,\|\, W_s.$$
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 $r' \cdot b^2 \equiv a^3 \pmod p$ 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 $n \cdot h$, 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$((\text{seedE} + i) \bmod 2^{160})$.
5. Let x_U be the bit string obtained by the concatenation of $x_0, x_1, \ldots, x_s$ as follows:
$$x_U = x_0 \,\|\, x_1 \,\|\, \cdots \,\|\, x_s.$$
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. 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 $n \cdot h$, 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 $x_i = \text{SHA-1}((\text{seedE} + i) \bmod 2^{160})$.

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

$$x_U = x_0 \,\|\, x_1 \,\|\, \cdots \,\|\, x_s.$$

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 $P \neq P'$ or if $nP \neq \mathcal{O}$ then reject.

9. Accept.

3.3 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, $0 \leq l \leq n - 1$, 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 than 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.

3.4 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. These estimates are for software implementations; we do not assume that any special hardware for parallelized Pollard rho attacks is used.

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 together with some rho-method specific operations such as evaluations of hash functions and/or a membership test. Also recall from Section 2.2 that for Koblitz curves over $\mathbb{F}_{2^m}$, the number of iterations can be reduced up to a factor of $\sqrt{2m}$, and for all other curves up to a factor of $\sqrt{2}$.

Thus, if a computer can perform l operations per second, then the number of computer days required before a discrete logarithm is found is expected to be roughly

$$\frac{1}{l \times 60 \times 60 \times 24} \times \frac{\sqrt{\pi n}}{2\sqrt{m}M} \approx 10^{-5} \times \frac{\sqrt{n}}{l\sqrt{m}M} \text{ machine days}$$

in the case of Koblitz curves over $\mathbb{F}_{2^m}$, and

$$\frac{1}{l \times 60 \times 60 \times 24} \times \frac{\sqrt{\pi n}}{2M} \approx 10^{-5} \times \frac{\sqrt{n}}{lM} \text{ machine days}$$

for all other curves.

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, may perform 16000 iterations per second for a curve over $\mathbb{F}_{2^{89}}$. Thus such an implementation would require

$$10^{-5} \times \frac{\sqrt{2^{89}}}{16000 \times M} \approx \frac{15550}{M} \text{ machine days}$$

to find a single discrete logarithm. So, for example, one such machine running 24 hours a day would require 15550 days. A network of 3000 such machines would require about 5 days.

When estimating the challenge problems, we also take into account that the iterations scale quadratically on the number of machine words required by the field. We assume that we work with a 32-bit machine. Then, for example, a 109-bit field requires 4 machine words while a 89-bit field requires only 3 machine words. This means that each iteration in a 109-bit field should cost $(4/3)^2$ as much. Hence, a Pentium 100 can perform about $l = (3/4)^2 \times 16000 = 9000$ iterations per second for a curve over $\mathbb{F}_{2^{109}}$.

For a curve over a 89-bit prime field, we estimate that a Pentium 100 can perform about $l = 48000$ iterations per second. For a Koblitz curve over $\mathbb{F}_{2^{89}}$, we estimate that a Pentium 100 can perform about $l = 24000$ iterations per second. Here we assume that iterations are done on orbits of points rather that on points. This slightly increases the time needed for one iteration while it considerably reduces the expected number of iterations.

The 109-bit Level I challenges are feasible using a very large network of computers, and have now been solved. The 131-bit Level I challenges will be require significantly more work, but may be within reach.[4]

The Level II challenges are infeasible given today's computer technology and knowledge, with the possible exception of the 163-bit challenges, which according to NIST recommendations are unsuitable for use beyond 2010, and as such may be feasible. The elliptic curves for these challenges meet the stringent security requirements imposed by existing and forthcoming ANSI banking standards [X962, X963].

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

The estimates are included in the tables of Section 6.2.

4 Exercise Lists and Challenge Lists

4.1 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 a matter of days, and the 97-bit exercise in a matter of weeks using a network of 3000 computers.

[4]This sentence was formerly "The 131-bit Level I challenges are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered." which contradicted former sentences in the abstract. Certicom thanks Dan Bernstein for alerting us to this inconsistency.

The 109-bit Level I challenges are feasible using a very large network of computers, and have now been solved. The 131-bit Level I challenges will be require significantly more work, but may be within reach.[5]

The Level II challenges are infeasible given today's computer technology and knowledge, with the possible exception of the 163-bit challenges, which according to NIST recommendations are unsuitable for use beyond 2010, and as such may feasible. The elliptic curves for these challenges meet the stringent security requirements imposed by existing and forthcoming ANSI banking standards [X962, X963].

4.1.1 Exercises

Exercise	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2-79	79	352	*Handbook of Applied Cryptography & Maple V Software*
ECC2-89	89	11278	*Handbook of Applied Cryptography & Maple V Software*
ECC2K-95	97	18322	$ 5,000
ECC2-97	97	180448	$ 5,000

4.1.2 Level I challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2K-108	109	1.3×10^6	$10,000
ECC2-109	109	2.1×10^7	$10,000
ECC2K-130	131	2.7×10^9	$20,000
ECC2-131	131	6.6×10^{10}	$20,000

4.1.3 Level II challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECC2K-163	163	2.48×10^{15}	$30,000
ECC2-163	163	2.48×10^{15}	$30,000
ECC2-191	191	4.07×10^{19}	$40,000
ECC2K-238	239	6.83×10^{26}	$50,000
ECC2-238	239	6.83×10^{26}	$50,000
ECC2K-358	359	7.88×10^{44}	$100,000
ECC2-353	359	7.88×10^{44}	$100,000

[5]This sentence was formerly "The 131-bit Level I challenges are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered." which contradicted former sentences in the abstract. Certicom thanks Dan Bernstein for alerting us to this inconsistency.

4.2 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 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 challenges are feasible using a very large network of computers, and have now been solved. The 131-bit Level I challenges will be require significantly more work, but may be within reach.[6]

The Level II challenges are infeasible given today's computer technology and knowledge, with the possible exception of the 163-bit challenges, which according to NIST recommendations are unsuitable for use beyond 2010, and as such may feasible. The elliptic curves for these challenges meet the stringent security requirements imposed by existing and forthcoming ANSI banking standards [X962, X963].

4.2.1 Exercises

Exercise	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-79	79	146	*Handbook of Applied Cryptography & Maple V Software*
ECCp-89	89	4360	*Handbook of Applied Cryptography & Maple V Software*
ECCp-97	97	71982	$ 5,000

4.2.2 Level I challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-109	109	9.0×10^6	$10,000
ECCp-131	131	2.3×10^{10}	$20,000

4.2.3 Level II challenges

Challenge	Field size (in bits)	Estimated number of machine days	Prize (US$)
ECCp-163	163	2.3×10^{15}	$30,000
ECCp-191	192	4.8×10^{19}	$40,000
ECCp-239	239	1.4×10^{27}	$50,000
ECCp-359	359	3.7×10^{45}	$100,000

[6]This sentence was formerly "The 131-bit Level I challenges are expected to be infeasible against realistic software and hardware attacks, unless of course, a new algorithm for the ECDLP is discovered." which contradicted former sentences in the abstract. Certicom thanks Dan Bernstein for alerting us to this inconsistency.

5 Challenge Rules

5.1 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.
5520 Explorer Drive
Mississauga, Ontario
Canada L4W 5L1

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`.

5.1.1 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.

5.1.2 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 elliptic curves over the finite field $\mathbb{F}_{2^m}$ and one elliptic curves over the finite field $\mathbb{F}_p$. 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 $\mathbb{F}_{2^m}$), the ECC2K-97 exercise (97-bit field size over the field $\mathbb{F}_p$) would still be available to solve and the cash prize available for award to the person(s) with the correct solution.

6 Status

The challenge was released on November 6, 1997, at 1 p.m. EST. The table below shows which problems have been solved so far. Here, the date given as end date indicates the day of submission of the solution to Certicom.

Challenge	End Date	Elliptic Curve Operations	Iterations per Second	Machine Days
ECC2-79	Dec. 16, 1997	1.7×10^{12}	170000	116
ECC2-89	Feb. 9, 1998	1.8×10^{13}	187000	1114
ECC2K-95	May 21, 1998	2.2×10^{13}	149000	1709
ECC2-97	Sep. 22, 1999	1.2×10^{14}	227000	6118
ECC2K-108	Apr. 4, 2000	2.3×10^{15}	160364	166000
ECC2-109	Apr. 8, 2004			
ECCp-79	Dec. 6, 1997	1.4×10^{12}	314000	52
ECCp-89	Jan. 12, 1998	2.4×10^{13}	388000	716
ECCp-97	Mar. 18, 1998	2.0×10^{14}	361000	6412
ECCp-109	Oct. 15, 2002	3.6×10^{16}		

Since the algorithms for all problems were based on Pollard's rho method [Pollard], the number of elliptic curve operations indicated above is the same as the number of iterations in the rho method.

The fourth column of the table indicates how many such iterations per seconds were performed by the challenge solvers' routines on a 500 MHz Alpha workstation, which is a typical machine used in the computations. Here, any speedup going from smaller to larger field sizes is due to code optimization. The penultimate column shows how many machine days on a 500 MHz Alpha would have been necessary if the whole computation had been performed on a single such machine running 24 hours a day.

All problems except the ECCp-97 problem were solved using the parallelized Pollard rho method due to van Oorschot and Wiener [VW]; for the ECCp-97 problem, a parallelized Brent-type cycle-finding algorithm was used. Furthermore, for the ECC2K-95 and ECC2K-108 problem the number of iterations was reduced since the iterations were performed on orbits rather than on individual points.

For an implementation report of the solution of some of the exercises, see [Escott].

6.1 Details of the solved problems

6.1.1 ECC2-79

- Solution: $\log_P Q =$ 3AA0 68A09F1E D21E2582
- Date: 16 Dec 1997
- Communicated by: Robert J. Harley (`Robert.Harley@inria.fr`)
- Who solved it: group of about 20 people
- Method: parallelized Pollard rho method:
  - 1 737 410 165 382 iterations
  - 1 617 distinguished points

6.1.2 ECC2-89

- Solution: $\log_P Q = $ 22A2A8 8267271F 48147AA5
- Date: 9 Feb 1998
- Communicated by: Robert J. Harley (Robert.Harley@inria.fr)
- Who solved it: group of about 70 people
- Method: parallelized Pollard rho method:
  - 18 161 819 582 507 iterations
  - 17 543 distinguished points

6.1.3 ECC2K-95

- Solution: $\log_P Q = $ 7A4249BA 547313AF 5D482092
- Date: 21 May 1998
- Communicated by: Robert J. Harley (Robert.Harley@inria.fr)
- Who solved it: group of about 47 people
- Method: parallelized Pollard rho method, with speedup by introduction of equivalence classes:
  - Roughly 21 600 000 000 000 iterations
  - 74 268 distinguished points
  - equivalence classes: the orbits under the involution $[-1]$ and the Frobenius endomorphism $[(-1+\sqrt{-7})/2]$ (= clusters of 194 points).

6.1.4 ECC2-97

- Solution: $\log_P Q = $ CF765639 D6B81943 FD89D60F
- Date: 22 Sep 1999
- Communicated by: Robert J. Harley (Robert.Harley@inria.fr)
- Who solved it: group of about 195 people
- Method: parallelized Pollard rho method:
  - 119 248 522 782 547 iterations
  - 127 492 distinguished points

6.1.5 ECC2K-108

- Solution: $\log_P Q = $ 0256 F98379F7 214306BE 5BE678FD
- Date: 4 Apr 2000
- Communicated by: Robert J. Harley (Robert.Harley@inria.fr)
- Who solved it: group of about 1300 people
- Method: parallelized Pollard rho method with speedup:
  - 2.3×10^{15} iterations
  - 2.05 million distinguished points

6.1.6 ECC2-109

- Solution: $\log_P Q = $ 0F1E 0ADD3449 596419C3 59DBDB7E
- Date: 8 April 2004
- Communicated by: Chris Monico (cmonico@koch.math.ttu.edu)
- Who solved it: group of about 2600 users
- Method: parallelized Pollard rho method [VW], with ideas of Edlyn Teske for getting walks more closely approximating a random walk
  - 20 million distinguished points
  - Gross CPU times roughly equivalent Athlon XP 3200+ working nonstop for about 1200 years
  - 17 months of calendar time.

6.1.7 ECCp-79

- Solution: $\log_P Q = $ 1387 56822DD5 FB093766
- Date: 6 Dec 1997
- Communicated by: Robert J. Harley (Robert.Harley@inria.fr)
- Who solved it: W. Baisley and R.J. Harley
- Method: parallelized Pollard rho method, parallelized Brent-type cycle-finding algorithm:
  - 1 400 000 000 000 iterations

6.1.8 ECCp-89

- Solution: $\log_P Q =$ 0113C284 D9BD7B58 BCA30C67
- Date: 12 Jan 1998
- Communicated by: Robert J. Harley (`Robert.Harley@inria.fr`)
- Who solved it: group of about 57 people
- Method: parallelized Pollard rho method:
  - 24 249 418 904 337 iterations
  - 36 345 distinguished points

6.1.9 ECCp-97

- Solution: $\log_P Q =$ 01 6C86AA7C ACF69F1D D28B3E2F
- Date: 18 Mar 1998
- Communicated by: Dimitris Tsapakidis (`dimitris@alien.bt.co.uk`)
- Who solved it: group of about 588 people
- Method: parallelized Pollard rho method:
  - 2.0×10^{14} iterations
  - 186 364 distinguished points

6.1.10 ECCp-109

- Solution: $\log_P Q =$ 281183840311601949668207954530684
- Date: 15 October 2002
- Communicated by: Chris Monico (`cmonico@koch.math.ttu.edu`)
- Who solved it: group of about 10308 users
- Method: parallelized Pollard rho method
  - 68228557 distinguished points
  - 549 days of calendar time

6.2 Status and estimates of ECC Challenge problems

Here we list the tables of both solved and unsolved problems, their status and Certicom's estimate of computation required for solution.

For a detailed explanation of the estimates, please refer to Section 3.4.

6.2.1 Elliptic Curves over $\mathbb{F}_{2^m}$

Exercises						
				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECC2-79	Nov. 6, 1997	Dec. 16, 1997	1.7×10^{12}	4.9×10^{11}	16000	352
ECC2-89	Nov. 6, 1997	Feb. 16, 1998	1.8×10^{13}	1.6×10^{13}	16000	11278
ECC2-79	Nov. 6, 1997	May 21, 1998	2.6×10^{13}	1.8×10^{13}	24000	8637
ECC2-79	Nov. 6, 1997	Sep. 22, 1999	1.2×10^{14}	2.5×10^{14}	16000	180448

Exercises						
				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECC2-109	Nov. 6, 1997	Apr. 8, 2004		1.6×10^{16}	9000	2.1×10^{7}
ECC2K-108	Nov. 6, 1997	Apr 4, 2000	2.3×10^{15}	1.5×10^{15}	13500	1.3×10^{6}
ECC2-131	Nov. 6, 1997			3.3×10^{19}	5760	6.6×10^{10}
ECC2K-130	Nov. 6, 1997			2.0×10^{18}	8640	2.7×10^{9}

Exercises						
				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECC2-163	Nov. 6, 1997			2.1×10^{24}	4000	6.2×10^{15}
ECC2K-163	Nov. 6, 1997			1.7×10^{23}	6000	3.2×10^{14}
ECC2-191	Nov. 6, 1997			3.5×10^{28}	4000	1.0×10^{20}
ECC2-238	Nov. 6, 1997			4.2×10^{35}	2250	2.1×10^{27}
ECC2K-238	Nov. 6, 1997			2.7×10^{34}	3357	9.2×10^{25}
ECC2-353	Nov. 6, 1997			1.1×10^{53}	1000	1.3×10^{45}
ECC2K-358	Nov. 6, 1997			3.6×10^{52}	1500	2.8×10^{44}

6.2.2 Elliptic Curves over $\mathbb{F}_p$

Exercises

				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECCp-79	Nov. 6, 1997	Dec. 6, 1997	1.4×10^{11}	6.1×10^{11}	48000	146
ECCp-89	Nov. 6, 1997	Jan. 12, 1998	2.4×10^{13}	1.8×10^{13}	48000	4360
ECCp-97	Nov. 6, 1997	Mar. 18, 1998	2.0×10^{14}	3.0×10^{14}	48000	71982

Exercises

				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECCp-109	Nov. 6, 1997	Oct. 15 2002	3.6×10^{16}	2.1×10^{16}	27000	9.0×10^{6}
ECCp-131	Nov. 6, 1997			3.5×10^{19}	17820	2.3×10^{11}

Exercises

				Estimated number of		
Challenge	Start Date	End Date	Number of Elliptic Curve Operations Taken	Elliptic Curve Operations	Iterations per second	Machine days
ECCp-163	Nov. 6, 1997			2.4×10^{24}	12000	2.3×10^{15}
ECCp-191	Nov. 6, 1997			4.9×10^{28}	12000	4.8×10^{19}
ECCp-239	Nov. 6, 1997			8.2×10^{35}	6750	1.4×10^{27}
ECCp-359	Nov. 6, 1997			9.6×10^{53}	3000	3.7×10^{45}

References

[BK] 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), 114–145.

[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`

[Diem1] C. Diem, "On the discrete logarithm problem in class groups of curves", *Mathematics of Computation*, to appear.

[Diem2] C. Diem, "On the discrete logarithm problem in elliptic curves", preprint, 2009.

[Escott] 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/

[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.

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

[Gaudry] P. Gaudry, "Index calculus for abelian varieties and the elliptic curve discrete logarithm problem", *Journal of Symbolic Computation*, to appear.

[GHS] P. Gaudry, F. Hess and N. Smart, "Constructive and destructive facets of Weil descent on elliptic curves", *Journal of Cryptology*, 15 (2002), 19–46.

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

[GS] S. Galbraith and N. Smart, "A cryptographic application of Weil descent", *Codes and Cryptography*, Lecture Notes in Computer Science, 1746 (1999), Springer-Verlag, 865–874.

[Hess] F. Hess, "Generalising the GHS attack on the elliptic curve discrete logarithm problem", *LMS J. Comput. Math.* 7 (2004), 167–192.

[HMV] G. Harper, A. Menezes and S. Vanstone, "Public-key cryptosystems with very small key lengths", *Advances in Cryptology – EUROCRYPT '85*, Lecture Notes in Computer Science, volume 658, Springer-Verlag, pages 163–173, 1993.

[JKSST] M. Jacobson, N. Koblitz, J. Silverman, A. Stein and E. Teske, "Analysis of the xedni calculus attack", *Designs, Codes and Cryptography*, to appear, 2000. Also available at http://www.cacr.math.uwaterloo.ca/

[JMS] M. Jacobson, A. Menezes and A, Stein, "Solving elliptic curve discrete logarithm problems using Weil descent", *Journal of the Ramanujan Mathematical Society* 16 (2001), 231-260.

[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 Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

[MMT] M. Maurer, A. Menezes and E. Teske, "Analysis of the GHS Weil descent attack on the ECDLP over characteristic two finite fields of composite degree", *LMS Journal of Computation and Mathematics* 5 (2002), 127-174.

[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.

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

[MQ] A. Menezes and M. Qu, "Analysis of the Weil descent attack of Gaudry, Hess and Smart", *Topics in Cryptology—CT-RSA 2001*, Lecture Notes in Computer science, volume 2020 (2001), Springer-Verlag, 308–318.

[MT] A. Menezes and E. Teske, "Cryptographic implications of Hess' generalized GHS attack", *Applicable Algebra in Engineering, Communication and Computing* 16 (2006), 439–460.

[MTW] A. Menezes, E. Teske and A. Weng, "Weak fields for ECC", *Topics in Cryptology–CT-RSA 2004*, Lecture Notes in Computer Science, volume 2964 (2004), Springer-Verlag, 366–386.

[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.

[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.

[Semaev] 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.

[Silverman] J. Silverman, “The xedni calculus and the elliptic curve discrete logarithm problem”, *Designs, Codes and Cryptography*, to appear, 2000. Also available at http://www.cacr.math.uwaterloo.ca/

[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.

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

[SS2] 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.

[VW] P. van Oorschot and M. Wiener, “Parallel collision search with cryptanalytic applications”, *Journal of Cryptology*, 12 (1999), 1–28. (An earlier version appeared in the Proceedings of the 2nd ACM Conference on Computer and Communications Security, ACM Press, pages 210-218, 1994.)

[WZ] 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.

[X962] American National Standard X9.62-2005, “Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)”, 2005.

[X963] American National Standard X9.63-2001, “Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography”, 2001.

A Curve Details

Also available at http://www.certicom.com/index.php/curves-list

```
=======
ECC2-79
=======

m = 79

f = x^79 +x^9 + 1

seedE = D3E5D53A 4D696E67 68756151 757779B8 379AC409
```

```
a = 4A2E 38A8F66D 7F4C385F
b = 2C0B B31C6BEC C03D68A7

seedP = 50CBF1D9 5CA94D69 6E676875 615175F1 6A36A3B1
U_x = 5674 7BC3DDBF F399EF4B
U_y = 49FF 222B2065 008D3C2C
P_x = 30CB 127B63E4 2792F10F
P_y = 547B 2C88266B B04F713B

h = 02
n = 4000 00000045 31A2562B

seedQ = 4EC95934 D696E676 87561517 56D5D2A8 FCD02E68
V_x = 5B6E 06AFB573 C1CF2BF5
V_y = 69F9 74972600 1DB5B8D9
Q_x = 0020 2A9F0350 14497325
Q_y = 5175 A6485955 2F97C129

-------------------------------

=======
ECC2-89
=======

m = 89

f = x^89 + x^38 + 1

seedE = F1DCCA0F F8B74D69 6E676875 6151754B B21CD49C
a = 0095AA3E 660B75E7 7315E94E
b = 01AC2701 C6C54021 D1BF0A72

seedP = EE37E551 34D696E6 76875615 17502BE9 9063A9F7
U_x = 01A62F57 0A344A1F 13394F91
U_y = 001AD5A2 0035D7DE A948F256
P_x = 01675C1B C9BD2348 52E8123C
P_y = 00426B72 7619B0E9 3561C6A3

h = 02
n = 01000000 00000B41 C8C9D8FD

seedQ = FD26A4D6 96E67687 5615175E 8B00ABB8 D1EA4015
V_x = 008ECBE3 9E439C15 316623E7
V_y = 018FE41E 51BA7333 A52468F4
Q_x = 01BE721F 8E3B6DBE E40DCE98
```

```
Q_y = 004DD88A 4BDAD73E 0B32EBB3

---------------------------------------

=======
ECC2-97
=======

m = 97

f = x^97 +x^6 + 1

seedE = 2B354920 B724D696 E6768756 1517585B A1332DC8
a = 01 EA5CE2B7 F0A58E01 B4389418
b = 00 9687742B 6329E706 80231988

SeedP = 21BD8EAF 4D696E67 68756151 75BD8678 E00A46B2
U_x = 01 39CB479E F7D86F46 476E0663
U_y = 01 156ACA98 B0BC9ACF F52DDE55
P_x = 00 94D3BA57 43058882 62363929
P_y = 01 3CE0562C C51EF416 B2C0CA80

h = 02
n = 01 00000000 00007383 E2DE1E81

SeedQ = 315ADA96 4D696E67 68756151 7535FA24 CBF48397
V_x = 01 8E2B2CC8 5ECF1CC6 7FFA66C7
V_y = 00 BE53825D 9FF7AD1A 7C75E1E9
Q_x = 00 EBCEC4B5 961C75DF 04EB6AAE
Q_y = 01 3768154C 2131B67B 39A0339F

--------------------------

========
ECC2K-95
========

m = 97

f = x^97 +x^6 + 1

seedE = NO
a = 00 00000000 00000000 00000000
b = 00 00000000 00000000 00000001
```

```
seedP = 50CBF1D9 5CA94D69 6E676875 615175F1 6A36A3B8
U_x = 01 52891484 9B12FD64 D2FEFDEB
U_y = 00 491D7BBC 0AD80C4A B3CA9CA7
P_x = 00 8A84FB02 034F7771 DC940097
P_y = 01 D2F10A47 1D48A720 F18F6339

h = 04
n = 7FFFFFFF FFFFA9FE D6609AF7

seedQ = 4D696E67 68756151 75037272 26FEE987 E96A11FC
V_x = 00 C81592AD 4F662DD8 0FC42F72
V_y = 01 BCB151F0 F695639B B2B6FB04
Q_x = 00 E0BC08AC 5818F303 E2B05E90
Q_y = 01 34C028FC 3393124D 673E6F8E

-------------------------
Part 2. Level I challenges
-------------------------

========
ECC2-109
========

m = 109

f = x^109 +x^9 + x^2 +x +1

seedE = 4D696E67 68756151 75037272 26FEE987 E96A11FD
a = 14BA A8C4131E 992C7E35 FCF70CE3
b = 1333 BE219E61 625E4C2B 6B1032D9

seedP = 1DD8F2C7 84D696E6 76875615 175A36EF 3F72194E
U_x = 1E22 5E8E554E 637C2DC8 E3C56B61
U_y = 04A3 4282B3B6 C22B985E 721134DD
P_x = 0D94 7CCA2312 E483059F 0917151C
P_y = 1C59 E6436972 07374CFE AB259AB6

h = 02
n = 1000 00000000 00053701 AB26100B

seedQ = A57C94DF 4D696E67 68756151 75E8B486 4BC74FFB
V_x = 03AF 701C0286 D6157981 A304B428
V_y = 1AC0 5F2A6251 93473DB5 F9906888
Q_x = 02DC 77273A0B 4961A85C 6B12027B
Q_y = 02DA 3FC33E70 5D4FD1CA D73D9E61
```

```
--------------------------------
========
ECC2K-108
========

m = 109

f = x^109 +x^9 + x^2 +x +1

seedE = NO
a = 0000 00000000 00000000 00000001
b = 0000 00000000 00000000 00000001

seedP = 078AB942 84D696E6 76875615 175A36EF 3F7298E2
U_x = 1170 08B979AF FA5985F9 90570ED8
U_y = 1DCA 7D761F1E 42453708 D0530BA6
P_x = 0478 C46CC963 38CED915 65E17257
P_y = 1E79 65E4A3AF B73A48FC 9AB790E9

h = 02
n = 0FFF FFFFFFFF FFA621B0 2C383E9B

seedQ = 176894DF 4D696E67 68756151 75E8B486 4BC74261
V_x = 05C5 9E717F6B 3D65D538 64F82F6D
V_y = 006A 19B5E3E2 BCFA2A5D BF6EC682
Q_x = 1FF0 CE5EC618 93F2119C 3077C59E
Q_y = 1F20 E9B010AC 691C9B87 B438241D

---------------------------------

========
ECC2-131
========

m = 131

f = x^131 + x^13 + x^2 + x + 1

seedE = 8950783D 4D696E67 68756151 750A6732 71EA86E3
a = 07 EBCB7EEC C296A1C4 A1A14F2C 9E44352E
b = 00 610B0A57 C73649AD 0093BDD6 22A61D81

seedP = B5555555 5554D696 E6768756 1517586A A17BF3F4
U_x = 01 DB221DEB 61DC208A DA4D8888 5A787969
```

```
U_y = 07 81B4EEEA 7DBDE2FE 9BE142DA E37BDF96
P_x:= 00 439CBC8D C73AA981 030D5BC5 7B331663
P_y:= 01 4904C07D 4F25A16C 2DE036D6 0B762BD4

h = 02
n = 04 00000000 00000002 6ABB991F E311FE83

seedQ= 19D059FF E6124D69 6E676875 61517565 06A34421
V_x = 00 03B50139 9F4A5D28 860EA3B5 6C6890C1
V_y = 00 AF1A352D D10974DB 899C6685 BDBB7CB6
Q_x = 06 02339C5D B0E9C694 AC890852 8C51C440
Q_y = 04 F7B99169 FA1A0F27 37813742 B1588CB8

---------------------------------------

========
ECC2K-130
========

m = 131

f = x^131 + x^13 + x^2 + x + 1

seedE = NO
a = 00 00000000 00000000 00000000 00000000
b = 00 00000000 00000000 00000000 00000001

seedP = 092FE1A8 9014D696 E6768756 1517586A A17BF123
U_x = 02 B8CB4816 38A7BB32 A5214816 621C9B9E
U_y = 07 CC4AAFC3 5046760A 6EF92D38 BFB9F5E1
P_x = 05 1C99BFA6 F18DE467 C80C23B9 8C7994AA
P_y = 04 2EA2D112 ECEC71FC F7E000D7 EFC978BD

h = 04
n = 2 00000000 00000000 4D4FDD57 03A3F269


seedQ= 328D0AE9 E6124D69 6E676875 61517565 06A34A25
V_x = 07 04AA2F3B 92953C63 B8CBB577 A6F83F07
V_y = 03 94249E7F 29B33ADE 47ABEE95 27EEE974
Q_x = 06 C997F3E7 F2C66A4A 5D2FDA13 756A37B1
Q_y = 04 A38D1182 9D32D347 BD0C0F58 4D546E9A

---------------------------------------
Part 3. Level II challenges
```

```
----------------------------------------

========
ECC2-163
========

m = 163

f = x^163 + x^8 + x^2 + x + 1

seedE = D2C0FB15 760860DE F1EEF4D6 96E67687 56151754
a = 02 5C4BEAC8 074B8C2D 9DF63AF9 1263EB82 29B3C967
b = 00 C9517D06 D5240D3C FF38C74B 20B6CD4D 6F9DD4D9

seedP = C368944D 696E6768 75615175 FF31C825 CC82534A
U_x = 04 342429E5 9B4E1052 222769E1 AB51C17A 53EAB862
U_y = 01 02FB92FE EB65AD06 8469D2DD 15BC0906 C9520891
P_x = 02 3A2E9990 4996E867 9B50FF1E 49ADD8BD 2388F387
P_y = 05 FCBFE409 8477C9D1 87EA1CF6 15C7E915 29E73BA2

h = 02
n = 04 00000000 00000000 0001E60F C8821CC7 4DAEAFC1

seedQ = DFD5F8E2 E38F4D69 6E676875 615175F3 B5115321
V_x = 03 85E70316 D171C67A C6C74463 9CCF27B9 7CDAFCC9
V_y = 06 D5323AEA D193FB57 BB37878A 46125B5A ACE1A5C2
Q_x = 04 38D8B382 1C8E9264 637F2FC7 4F8007B2 1210F0F2
Q_y = 07 3FCEA8D5 E247CE36 7368F006 EBD5B32F DF4286D2

----------------------
=========
ECC2K-163
=========

m = 163

f = x^163 + x^8 + x^2 + x + 1

seedE = NO
a = 00 00000000 00000000 00000000 00000000 00000001
a = 00 00000000 00000000 00000000 00000000 00000001

seedP = 4D696E67 68756151 75037272 26FEE987 E96A11FD
U_x = 01 D99596B0 2A3DF13E 000010DA 0469AB98 2D270F45
U_y = 06 8FB2A65B 990287F9 89830112 2AC0C5CD EECF3867
```

```
P_x = 02 091945E4 2080CD9C BCF14A71 07D8BC55 CDD65EA9
P_y = 06 33156938 33774294 A39CF6F8 C175D02B 8E6A5587

h = 02
n = 04 00000000 00000000 00020108 A2E0CC0D 99F8A5EF

seedQ = 6BA57515 8BDAF0D9 694D696E 67687561 517574BB
V_x = 00 085E74F0 3C80875A 74FC6BA6 72F879B2 0FE55A86
V_y = 06 0869E955 97DAFC58 5D6DC4F5 24954C91 E03D003F
Q_x = 00 7530EE86 4EDCF4A3 1C85AA17 C197FFF5 CAFECAE1
Q_y = 07 5DB1E80D 7C4A92C7 BBB79EAE 3EC545F8 A31CFA6B

----------------------

========
ECC2-191
========

m = 191

f = x^191 + x^9 + 1

seedE = 4E13CA54 2744D696 E6768756 1517552F 279A8C84
a = 2011D237 B30025CF D61F8D6F 66741CA6 CEDD313F 9BD4431F
b = 2E45EF57 1F00786F 67B0081B 9495A3D9 5462F5DE 0AA185EC

seedP = FD6F9C51 55B4D696 E6768756 15175BB2 BBBEB67B
U_x = 12D3B5BA 47EF83A9 A4FD6882 27CD6582 154EF5D6 BF7F536B
U_y = 0C2105BC E4329EE5 452B55B8 216CF453 945BE8BD F09A614C
P_x = 52F3E1EC FCF3B0A6 E0945D6B 92A8FFC4 2C280545 C0F8321B
P_y = 00F19F7A 172A1F71 91614DB1 DAE7CABC 01E85806 D42E2627

h = 02
n = 40000000 00000000 00000000 04A20E90 C39067C8 93BBB9A5

seedQ= 77733A8F C54D696E 67687561 517539F6 FEB6390C
V_x = 64BB3A91 0B9788FA D747A07E 1CDA1554 2D871430 8429678A
V_y = 3C81DE80 3F17F3F5 71681832 F7AFDE8A 3F737C07 473A677B
Q_x = 7911BC30 0972E2D3 385D6D52 8440710D D73F2814 72B961E3
Q_y = 492A2DED 9E740B36 03F39DAB AA0070F9 9674C57F 55AD4F1D

----------------------------

========
ECC2-238
```

```
========

m = 239

f = x^239 + x^36 + 1

seedE = D34B9A4D 696E6768 75615175 CA71B920 BFEFB05D
a = 23AA CC83DE35 49BEAE1B 601EDA88 BF45517D 5ADFED43 7F71784B 5DD911DF
b = 7904 08F2EEDA F392B012 EDEFB339 2F30F432 7C0CA3F3 1FC383C4 22AA8C16

seedP = F7930DE1 C514D696 E6768756 15175155 35EAEA7E
U_x = 707A FBF8A455 7F55BFAF 9857CC61 CAF4F1AF FF63BFAC D850B85C A5294FAC
U_y = 11B2 84BC6B71 D9BAEDEA F9DBB18D C9D05693 625AC030 37BE32CF E5A2292B
P_x = 4BC6 30D209D9 8B810FA4 2A95FDB7 7F87DBE6 8CC07611 0AADE0B2 AFC47B2B
P_y = 194C 8DA12F27 63E3E9AA 84F4F17D 56FF10FC F20F300F D1278FFB 976A91C2

h = 04
n = 2000 00000000 00000000 00000000 000F4D42 FFE1492A 4993F1CA D666E447

seedQ = 315ADA96 4D696E67 68756151 7535FA24 CBF48396
V_x = 68D5 EAFAE6E5 F00DDD65 28140D7F 74BD3C31 8E2B2CC8 5ECF1CC6 7FFA66C7
V_y = 324F 34B5E3AE F614BE7E A574AD85 29F811E0 D671B120 7A5488F6 FF01D308
Q_x = 2696 6E5DDE49 2AF6CEDA F8BB8BCA DA02BFEF 2944815F 142F5745 0CA145BF
Q_y = 4F2A C9ED2988 7CB53ACB B29AD031 B4A24C2C FA8A32D5 B164E89F A283966B

-----------------------------------------------------------

=========
ECC2K-238
=========

m = 239

f = x^239 + x^36 + 1

seedE = NO
a = 0000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
b = 0000 00000000 00000000 00000000 00000000 00000000 00000000 00000001

seedP = D9037BB3 2E4D696E 67687561 51757738 894AE594
U_x = 2D80 5D2B1A98 A6E5F31E 8D5C82E5 8868C312 D185EF19 639381A7 BDF35AA4
U_y = 63E3 3A720145 A553E4EC 5D739E8C 34948694 F459F3EA B028FC10 5A3953D8
P_x = 78CF AE9860F9 6A9D4826 1B7A9CF3 C5B94190 888B184B C3F83B41 6B56992E
P_y = 2DEB A096F37D EDBE7D13 C09F6C03 680B2E53 71A08FEC 01E37775 E31CC7C9
```

```
h = 04
n = 2000 00000000 00000000 00000000 005A79FE C67CB6E9 1F1C1DA8 00E478A5

seedQ = 2AC709C3 E56AD814 D696E676 87561517 5551F2EC
V_x = 41CC C4B89D8C 0F75AC99 26C20BB3 D1A6BD99 25E1FD10 E2FB62AE D94F4234
V_y = 27AF DB90DB41 27A68684 A23B06C1 2405B4BD 70852DA1 DF02DE12 338965F0
Q_x = 721D D31C6766 532EC44C 73BCBF54 B2969400 FA6C4CC1 556E89AC 1FF8622A
Q_y = 347D 6F86DDDE DD7F138F C33BEEAB 4FFA029A 2EB2F011 A5BC8C64 A773B519

-----------------------------------------------------------

========
ECC2-353
========

m = 359

f = x^359 +x^68 + 1

seedE = 2B354920 B724D696 E6768756 1517585B A1332DC6
a =       69 62E67E43 FB6D66C4 6C7FDDF5 A22E2C88 57537C42 8FD691A2 6D2F9BAE
    B871C276 DFA9928F 01634B14 59CF431E
b=       24 72E2D019 7C49363F 1FE7F5B6 DB075D52 B6947D13 5D8CA445 805D39BC
   34562608 9687742B 6329E706 80231988


seedP = 1DD8F2C7 84D696E6 76875615 175A36EF 3F721942
U_x =       1E 67C60608 D697AD3D DF21D09E D9A86087 84589D80 1C35E267 BEF6579C
      D973C116 0749649F 0B5484A2 3EB35A36
U_y =       68 DD8E6E25 BB36DA06 1B2CCEE9 9B5E50B0 E83C33A4 D28B390E 0945F51E
      6EC43DC7 3D7C9D7C F865457A E12E2727
P_x =       0E F6A87D27 D25EA944 A629664A 9D34664C 6A79DD86 360D8AB5 7FAE11DF
      30CB6E78 C7E01BA3 ADC081FA 76048092

P_y =       1D A4ED7191 74E356FC 0F968E80 D2974965 BB90805D BB1D8FE5 7333DFA5
      0D163B88 465BB5FB 55046F94 9AE27F51

h = 4C
n =       01 AF286BCA 1AF286BC A1AF286B CA1AF286 BCA1AF28 6BC9FB8F 6B85C556
    892C20A7 EB964FE7 719E74F4 90758D3B

seedQ = A57C94DF 4D696E67 68756151 75E8B486 4BC74FF8
V_x =       7B 2ECC36E3 29F399AC 0E84A382 813BB893 C73F9AD0 47345084 383F8164
      041C227D 1B167ADD 9E706C6A 4E708A01
V_y =       58 B14AE739 5186A895 31F77B8F 71DD6319 94034E42 F55A51C3 ECE7A8F8
```

```
      6CFFC738 F069A273 B7517233 1D2F74D4
Q_x =       4B D3C6E97B FF49C1B3 FD667684 834FBE8D 17E22F94 F795480B D5166B04
      A267F940 33D61FBA 43FDF627 58982462
Q_y =       60 5F4BC822 1F266F72 4DEF7500 69B240A6 7AC4ADAC 15386354 D7E1700C
      5F5F3FB6 769F2613 0D73A8D7 21A019E4

-------------------------------------------------------

=========
ECC2K-358
=========

m = 359

f = x^359 +x^68 + 1

seedE = NO
a =      00 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000001
b =      00 00000000 00000000 00000000 00000000 00000000 00000000 00000000
   00000000 00000000 00000000 00000001

seedP = 91CB22C7 84D696E6 76875615 175A36EF 670D2FB1
U_x =    41 F4092AEC F6C9A286 DA018DA1 0BC55D77 8EA9D74E 3BA352C4 940A7BA4
   1FF722B2 6022FB8F 3FE87E17 C01D4486
U_y =    15 49EBB859 7B6B122F ECAC9DD0 4E5C0ACF B368014A 3440CE6F 61EA808B
   68EA7EF1 04C957F5 339C2006 09EF79F5
P_x =    70 35ADDF5E AAB0CFBA 95883326 03F870D1 1AA41A6B 57ADA3A1 CA18F9C7
   169185A1 F191E962 93EE6E04 1C8E918C
P_y =    71 22A49D2E F45E68D1 1C0C0ED8 6B8AE422 F54E49BD A0AC1517 7F3824D3
   5809AD4F AEF7B7AB 4810E1DB EC5ABFF0

h =   02
n =      3F FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFF6F614 06970B03
   AE00FB51 2716A415 1CA0EAC9 D1B0E107

seedQ = 90FA2BDF 4D696E67 68756151 75E8B486 4BC74AA5
V_x =    44 430DFBFD B5E030CF D74C2ACB 4FC1A4D4 F48DF8A0 0AA1F43A 0B9A1561
   9C96CBED 7AC32BD9 630DC0C8 F455B93D
V_y =    69 1E83DD42 43FED6ED 2E4F5193 59870438 817A854D 78624E93 A7F20CCB
   E9FECA9E 2DA72B5A 07F04015 EB365E69
Q_x =    0F E3828D45 A36EC0A9 D04BB8C0 3586A41E D098F4BC 55718A76 04A005DB
   050E58F6 DC2A451C B9D83448 D7B1DDA6
Q_y =    53 22AFB6DA 476F2B9C 3ABF31C2 2978C318 AE358F6B 3373CEAC 599189A3
   5DD6C121 C23170D7 10563F43 6CAAFDC8
```

```
=============================================================================

=======
ECCp-79
=======

p = 62CE 5177412A CA899CF5

seedE = 3409C5C7 E50FF4D6 96E67687 56151755 2DF2B489
r = 1CE4 AF36EED8 DE22B99D
a = 39C9 5E6DDDB1 BC45733C
b = 1F16 D880E89D 5A1C0ED1

h = 01
n = 62CE 5177407B 7258DC31

seedP = 1FD36F3C 978D0398 EAB24D69 6E676875 61517593
P_x = 315D 4B201C20 8475057D
P_y = 035F 3DF5AB37 0252450A

seedQ = 37A5B90A 4D696E67 68756151 75D52727 640CFCA6
Q_x = 0679 834CEFB7 215DC365
Q_y = 4084 BC50388C 4E6FDFAB

-------------------------------------
=======
ECCp-89
=======

p = 0158685C 903F1643 908BA955

seedE = A9ACF4D6 96E67687 5615175C 32C58C7B 47CDBBD7
r = 00C8AE4F 7DE8918A A9FAB226
a = 0C8AE4F7 DE8918AA 9FAB2260
b = 00647E7E A1062AE6 9A7D1037

h = 01
n = 0158685C 903EF906 D7F58D47

seedP = A7663810 665AEF4D 696E6768 75615175 EAF5E109
P_x = 00C031D8 75DBF8E6 0BE95B0A
P_y = 0006F82C 1F879745 BF676D0A

seedQ = 98BCE9AF 94D696E6 7687561517592EC57621252B
```

```
Q_x = 00DE1AA9 4FF94DB6 4E763E2D
Q_y = 002A44C4 C2D4EE27 FA0A4BA9

--------------------------------------
=======
ECCp-97
=======
p = 01 6EA1595E D21AE4D8 D8420E35

seedE = C85CB697 1A944D69 6E676875 61517542 7A522231
r = 00 32AB225F 544F8CB6 9CDF219B
a = 00 47370916 A603B076 57C305C4
b = 01 124DF86D 04064F50 3D9925AF

h = 01
n = 01 6EA1595E D21AE98F B6CCA20D

seedP = 2A8A34D6 96E67687 56151750 19818262 2266F335
x = 00 D5D9E9DF F58A9232 A2749EBC
y = 01 1B34AE5A AB7C7AE5 5D6ABDB5

seedQ = DF7374D6 96E67687 56151756 01CEEABC 0C476F92
x = 00 DF7E84C4 2FEF50C5 316C508A
y = 00 F259BC58 3729DA0F E8B97336

-----------------------------------------
========
ECCp-109
========

p = 1BD5 79792B38 0B5B521E 6D9FB599

seedE = 1A4D696E 67687561 5175F8DC 47334A5E EDA1C34C
r = 0167 90D37BA1 CC98B1FE 53558DEA
a = 0FD4 C926FD17 8E9805E6 63021744
b = 153D 3CBB508F FE3A7F31 FF4FAFFD

h = 01
n = 1BD5 79792B38 0B049C4D 13A75AE5

seedP = 2409C5C7 E50FF4D6 96E67687 56151755 2DF2B48C
x = 04CC 974EBBCB FDC3636F EB9F11C7
y = 0761 1B0EB122 9C0BFC5F 35521692
```

```
seedQ = 2CC6F5DA 1DBA34D6 96E67687 5615175D 2D30BDF2
x = 0233 857E4E8B 5F005512 6E7D7B7C
y = 19C8 C91063EB 4276371D 68B6B4D9

-----------------------------------------------------------
========
ECCp-131
========

p = 04 8E1D43F2 93469E33 194C4318 6B3ABC0B

seedE = ECF764D6 96E67687 56151758 05744809 00C4742C
r = 01 F6CE5106 86622B77 6D651862 12A8E281
a = 04 1CB121CE 2B31F608 A76FC8F2 3D73CB66
b = 02 F74F717E 8DEC9099 1E5EA9B2 FF03DA58

h = 01
n = 04 8E1D43F2 93469E31 7F7ED728 F6B8E6F1

seedP = 232881DF CF57E4D6 96E67687 5615175D 990B21A7
x = 03 DF84A96B 5688EF57 4FA91A32 E197198A
y = 01 47211619 17A44FB7 B4626F36 F0942E71

seedQ = 2773E8F6 E4AE54D6 96E67687 56151755 2EA42393
x = 03 AA6F004F C62E2DA1 ED0BFB62 C3FFB568
y = 00 9C21C284 BA8A445B B2701BF5 5E3A67ED

------------------------------------------
========
ECCp-163
========

p = 05 177B8A2A 0FD6A4FF 55CDA06B 0924E125 F86CAD9B

seedE = 33855928 86AB4D69 6E676875 61517593 5F3CA3E1
r = 00 3FEA47B8 B292641C 57F9BF84 BAECDE8B B3ADCE30
a = 04 3182D283 FCE38807 30C9A2FD D3F60165 29A166AF
b = 02 0C61E945 9E53D887 1BCAADC2 DFC8AD52 25228035

h = 01
n = 05 177B8A2A 0FD6A4FF 55CCA7B8 A1E21C88 BD53B2C1


seedP = E52F9C29 66F3AC36 D35D24D6 96E67687 56151758
```

```
x = 00 17E70122 77E1B4E4 3F7BF746 57E8BE08 BACA175B
y = 00 AA03A0A8 26907046 97E8C504 CB135B2B 6EEF3C83

seedQ = 2E81EB8A D34E204D 696E6768 75615175 BC8A5026
x = 01 DC1E9A48 2085B3DF A722EB7A 541D5050 5ED31DCA
y = 01 2D71ECC1 578BFBE2 03D0C2CE 238EB606 0ADCAA1E

------------------------------------------
========
ECCp-191
========

p = 7DF5BB7B F830F63C 77667331 106F9001 B27D3994 1032F5E5

seedE = D154D696 E6768756 15175C7F 08129A53 5275A092
r = 21C0D807 AA07CF74 549AD58B 000F8D8F 829DBA7E 4551A0FC
a =  3BD4FDA0 0A3E52E1 AF5C9456 686AB1B9 6195810C 27C5B110
b =  24D1D433 1F8651B0 52E8042F A4325588 6E09BEF9 D3174872

h = 01
n = 7DF5BB7B F830F63C 77667331 5F125916 8CF99738 0ACA72C3

seedP = 17A5B90A 4D696E67 68756151 75D52727 640CFCA6
P_x = 3B511BC3 229CB4AE 654DFBC6 3210E278 3E91F43A A68D0EF4
P_y = 4619A505 395A031A 304C0B72 061099F3 D0840CA6 1DE2F4BC

seedQ = F14C7505 F2FD4284 D696E676 87561517 5D8675F1
Q_x = 1DA38EF4 CBA78B2C D1D31EB3 75BC9E19 34C62ACE D29C54EE
Q_y = 4F3CA5FF 71D32D54 72D7F9EC D39DEF45 517F3B87 6466C8F1

---------------------------
========
ECCp-239
========

p = 7CFB 4C973A86 CDAF8982 31E4960A CDBBF5B6 A9017DBE D75FFABD D892085D

seedE = 90298F17 F124D696 E6768756 15175C45 E53FC75E
r = 3D72 0B1FD4F9 2B7EDE22 71DFD09E 665EB8DE 732221C5 EE0DB16E 7D6125A7
a = 76D4 219CF749 8B5B471E 85BC4DAB A3CE47AD C806228F BB0BCE19 7C4F4556
b = 4F09 11A649B9 8CD0D3F6 95695E44 743EA948 E70B78CA B2C24C4E 7D50E2B3

h = 01
n = 7CFB 4C973A86 CDAF8982 31E4960A CCB3E442 837A1D55 1D28F3B4 95F5EC5F
```

```
seedP = F32881DF CF57E4D6 96E67687 5615175D 990B21A7
P_x = 0D35 ED464403 B23CC681 F18534C1 4B6FA2AD E7720523 F5094AD9 BFBE4752
P_y = 52F1 BC7C3C74 38A91099 FDD53666 A0185FB5 9688CA3E 38084090 3B589BEB

seedQ = E773E8F6 E4AE54D6 96E67687 56151755 2EA42392
Q_x = 2193 DCEAE32B C6EF6165 3DE4F1A1 41C15A9A 6A1A7296 802A887E BC0C7667
Q_y = 6429 7E89EE34 0CFF78A5 31998CC3 F3376AFD 3AE177DB E30B82C9 3045F79D

----------------------------------------------
========
ECCp-359
========
p =       58 D8420DF5 5D2B2000 FE2A55A0 32AB225F 544F8CB6 9CDF219B 0E394237
    21F32A19 9D58685C 903F1643 908BA969

seedE = C421F029 D47B02C4 D696E676 87561517 5ADC6719
r =       1A 1B3C30A7 B1A41DB7 F2981E99 7E492511 3E9754B4 E7B70774 A933B40E
    0D90FB36 3C73D50F 2233D1DA C98AB9C6
a =       08 77AEBB17 71A6EEA1 A7681809 B6884681 8D6434ED F6B4EF23 81672DE2
    CAE70CB1 BA3E6A5F BD6DE671 70E4FC62
b =       3A DE22E91F 88EC9316 5A5BA6F1 51AA1EF2 65FF5FD0 12F30B9A 2D12A0E2
    C3F5D7E6 95DDB2FA 75DE2139 E61D8DC8

h = 01
n =       58 D8420DF5 5D2B2000 FE2A55A0 32AB225F 544F8CB6 9CD0BE15 04766B9D
    D626631A 535BA1BA 6CB8D062 F94102ED

seedP = BE8DF98B 862664D6 96E67687 56151756 E8E60912
P_x =       2F 912B99AD 5D761593 C2CE9D24 54EE91EF D1C698A0 DA7C2EFE 0DB86964
      06885E63 EDB5CD29 C2735EC1 2183312D
P_y =       33 5E0C161B AB13BC46 DE0CD4E0 BA17913B 9C1EE26A 3DCF9022 DE774318
      96F329D8 283B3DC9 3C469564 F9043CAA


seedQ = B3814C05 0D44D696 E6768756 1517580C A4E29FF5
Q_x =       10 E3208F62 A90AE4AE F55EB0A7 1F733443 2AF091C5 E9D50461 70C9835E
      C1B92167 698DCD0B 8E9040BD C3AFA0B0
Q_y =       15 03887866 4A36573C 40D10B3F 5FCD999E E1B619BF A84614EF 172FEFD4
      949F188E 39BB40E1 A767A6DF 7458A13D

----------------------------------------------
```