

Elliptic Curve Cryptography

Dimitri Dimoulakis, Steve Jones, and Lee Haughton

May 05 2000

Abstract. Elliptic curves can provide methods of encryption that, in some cases, are faster and use smaller keys to provide an equivalent level of security. Like most successful encryption schemes, ECC uses a one way function to encrypt data. Cracking this encryption is similar to computing discrete logarithms, but different in many subtle ways, and considerably more difficult. This difference may be exploited in order to provide both increased speed and decreased key size for some given level of security.

In 1976, the Diffie-Hellman key exchange was invented. This technique has spawned numerous public-key cryptographic systems since then. These systems all have various strengths and weaknesses including:

- security
- key lengths
- signature sizes
- speed
- optimization for hardware/software
- complexity of implementation
- storage requirements
- industry/government standards
- patent coverage/licensing terms

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

1. Integer factorization problem: RSA and Rabin's Algorithm
2. Discrete logarithm problem: DSA, Diffie-Hellman key exchange, ElGamal
3. Elliptic curve discrete logarithm problem

None of these problems have been mathematically proven to be unbreakable. Rather, they are believed to be unbreakable because years of intensive study by leading mathematicians and computer scientists have failed to yield efficient algorithms for solving them. The best algorithms known for factorization and discrete logarithms take subexponential time. In contrast, the best algorithms known for elliptic curve discrete logarithms take fully exponential time. This means that as the problem size increases, the algorithms for solving elliptic curves become infeasible much more rapidly than the algorithms for factorization or discrete logarithms. For this reason, elliptic curve cryptosystems (ECC) offer equivalent security to systems such as RSA and DSA while using far smaller key sizes.

1 Mathematics of Elliptic Curves

Elliptic curves are defined as a combination of three things:

- The set of points (x, y) that satisfy the equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ where x, y , and the coefficients a_i are all from \mathbf{Z}_p^* (\mathbf{Z}_p^* is also written as $GF(p^n)$)
- A special point O called the "point at infinity."
- A special addition operator \oplus that adds these points together.

We say that the $m + 1$ points

$$E = O, (x_1, y_1), (x_2, y_2), \dots, (x_m, y_m) \quad (1)$$

are the points on the elliptic curve, and the group E has order $m + 1$. Within E , we can add two points together using the \oplus operator, so that $(x_i, y_i) \oplus (x_j, y_j) = C(x_k, y_k)$. In other words, points on an elliptic curve are considered a group. The special point O is the group's additive identity.

2 How to Pick a Good Elliptic Curve

A curve constructed from $GF(p^n)$, where p is a prime greater than 3 and n is usually 1, is known as a prime curve. Since prime curves are the best cryptographic curve family for software-based implementations, we chose prime curves for our program.

The elliptic curve equation $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ can be reduced to $y^2 = x^3 + Ax + B$, where $A, B \in GF(p^n)$ and $4A^3 + 27B^2 \neq 0$. So to generate the elliptic curve, all you need to do is choose values for A, B, p , and n . Naturally, it is not quite that easy. Not just any values of A, B, p , and n can be used. There are some weaknesses associated with certain values. For instance, if the order of the curve group happens to be exactly equal to p , the curve is anomalous and is vulnerable to subexponential discrete logarithm algorithms. Furthermore, if the curve is supersingular, or if the curve does not satisfy the Menezes-Okamoto-Vanstone (MOV) condition, it is not suitable for cryptography. Fortunately the probability of selecting an insecure curve is small, and it is a simple task to validate the security of a given curve.

3 Picking ECC Parameters

At the CRYPTO '98 conference, Daniel Bailey and Christof Paar suggested using primes of the form $p = 2^n \pm c$ and extension polynomials of the form $x^m - \omega$ to form the base field of the elliptic curve. The primary advantage of this method is that the base p modular reductions can be done rapidly using either the Mohan-Adiga algorithm or the following identity:

$$\sum_{i=0}^{2m-2} c_i x^i \equiv c_{m-1} x^{m-1} + \sum_{i=0}^{m-2} [\omega c_{m+i} + c_i] x^i \text{ mod } (x^m - \omega) \quad (2)$$

When selecting parameters, it is necessary to pick curves where it is easy to calculate the exact elliptic curve group order. The best algorithm for counting the elements of an arbitrary elliptic curve requires

$$O(\ln^8 n) \tag{3}$$

time due to the algorithm's complicated structure. Rather than using this time consuming process, we can use curves from either of two families to find the extension polynomial $\#E$. The first type of curve is one where p is small and $n > 1$ in $GF(p^n)$. In this case you can quickly count the curve's points by using the relation between the base field and the extension field. The second type of curve is one with a known complex multiplication discriminant. For example, CM3 curves have complex multiplication discriminant 3, which means there is a simple formula for counting the elements. The P1363 draft describes simple algorithms for finding the complex multiplication discriminant of many curves.

To make the encryption even faster, primes should be used for which Mohan-Adiga reduction is especially efficient.

The size of the prime factor of the group order is also of relevance. Since cracking ECC is on the same order of complexity as solving discrete logarithm problems, for data to remain secure for the next 20 years the effective key size of an algorithm should be at least 90 bits.

Many types of ECC are based on some existing encryption system. ElGamal is one such system. The ECC analog of ElGamal can add strength and speed to an already powerful encryption system. To encrypt with the ECC analog of ElGamal, you start out with a fixed publicly known finite field \mathbf{F}_q , elliptic curve E defined over it, and base point $B \in E$. Each user chooses a random integer a , which is kept secret, and computes and publishes the point aB . To send a message P_m to Bob, Alice chooses a random integer k and sends the pair of points

$$(kB, P_m + k(a_B B)) \tag{4}$$

(where $a_B B$ is Bob's public key). To read the message, Bob multiplies the first point in the pair by his secret a_B and subtracts the result from the second point:

$$P_m + k(a_B B) - a_B(kB) = P_m \tag{5}$$

Therefore, Alice sends a disguised P_m along with a "clue" k_B which is enough to remove the "mask" $ka_B B$ if one knows the secret integer a_B . If Eve were

to intercept the message, she could determine a_B from the publicly known information of B and $a_B B$, but she would first have to solve the discrete log problem on E which we will assume that she can't do.

The selection of (E, B) is done easily using the following method.

1. First let x, y, a be three random elements in the field.
2. Set $b = y^2 - (x^3 + ax)$.
3. Verify that $x^3 + ax + b$ does not have multiple roots. This is equivalent to $4a^3 + 27b^2 \neq 0$. If this condition is not met make another random choice of x, y, a .
4. Set $B = (x, y)$.
5. Now B is a point on the elliptic curve $y^2 = x^3 + ax + b$.

Although the number of points (N) is not necessary to know for ElGamal, it is a good idea to determine this anyway since it will verify the security. N should have a large prime factor. If this is not the case and N is a product of small primes, then the Pohlig-Silver-Hellman method can be used to solve the discrete logarithm problem more easily. So to find N

Our encryption is done using 20 byte blocks so that we could perform the operations on one word units. We used a base field of 136 bits. We used a customized data structure VLONG to hold the numbers since they are of a substantially large size. Everyone will use the same elliptic curve.

4 Our Implementation

In our original project design submission, we proposed doing a file encryptor/decryptor using ECC. Not only have we successfully implemented this, but we have combined it with an e-mail client and server of our own design that uses the ElGamal analog of ECC. We feel that these extra features will make our program more useful and relevant to the average computer user. Not only that, but an e-mail implementation would better make use of the public key properties of ECC. Our mail server not only holds the mail for each user, but it also holds the keys and other EC parameters for each group of users. This feature will make the encryption overhead on each message significantly smaller. Our programs were all tested over a TCP/IP network connection.

5 Design and Components

5.1 To compile:

Use the provided makefile. This program was developed under Linux on Intel machines, however, other systems may require library linking. If any of the targets containing tcnode won't compile, try using `-lnsl -lsocket` as additional parameters to `g++`;

5.2 Readecc

— Decrypts a received message.

Usage: `readecc ciphertextfile outputfile`

User specifies an encrypted file `ciphertext` and a plaintext file `outputfile` to which it should be translated.

5.3 Sendecc

— Encrypts an alphanumeric message and sends it to a server running in the specified hostname, where it is spooled (while encrypted).

Usage: `sendecc filename recipient@hostname reply-to@hostname`

User specified the file to encrypt, `filename`, the recipient (a valid e-mail address where an eccserver is running), and a `reply-to` address.

5.4 Keygen

— Generates a pair of public-key elliptic curve keys for use by the user who is currently logged in.

Usage: `keygen`

This program will ask the operating system which user is currently logged in, and generate a file called `username.key` containing a public key and a private key. This *-must-* be executed by the user before they can send or receive any messages.

5.5 Eccserver

— A daemon-like program that receives incoming encrypted files and spools them for later reading.

Usage: eccserver

This program should be run in the user's home directory or a subdirectory. This directory should contain a username.key file. It will output incoming messages (encrypted) using random filenames of the form usernameXXXXXX. These files can be read using readecc.

This paper was written and compiled with L^AT_EX. We felt that was the best option due to its ability to display the complex mathematical equations appearing in this document. Our work was split evenly, with all group members doing an equal amount of work. Steve and Lee worked primarily on the programming, and Dimitri worked primarily on the paper, but naturally the borders are very fuzzy; we all helped in some way with every part of the project, and all of our research was done as a group.

5.6 Known bugs:

- Since our encryption is done in 20 byte blocks the last three bytes come through as plaintext. These bytes may or may not be in order. We tried for hours to work out this bug, but were unable to determine the source of it. It seems like they should be encrypted since everything else is, but for some reason they are not. We think it must have something to do with the VLONG or our choice of curve, but we do not know for sure.
- The other minor problem is that it does not run as fast as we thought it would. Our expectations were somewhat high due to all the praise we have seen in numerous publications commenting on the system's speed as compared to other successful encryption systems. We did not determine the difference to be that significant.
- Our overhead was also a minor factor since we had to store 21 bytes for ever 20 bytes. Since not all plaintext mapped to a valid point on the curve, we adjusted for this by "shifting" the values or adding to them until they worked as valid VLONG's.

6 Conclusions

Software-based prime fields elliptic curve cryptosystems seem to be more efficient than traditional cryptosystems at high levels of security. The small size

of their private and public keys are a definite advantage, but this advantage only holds if each user in the group uses the same EC parameters. If the EC parameters need to be specified in each message, then the EC key size will be much larger.

7 References:

The implementation of VLONG's and curve operations is based on George Barwood's program NCTEST at:

<http://ds.dial.pipex.com/george.barwood/crypto.htm>

Koblitz, Neal. Introduction to Elliptic Curves and Modular Forms. Springer-Verlag, 1993.

Koblitz, Neal. Algebraic Aspects of Cryptography, Second Edition. Springer-Verlag, 1994.

Koblitz, Neal. A Course on Number Theory and Cryptography, Second Edition. Springer-Verlag, 1994.

Menezes, Alfred. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publications, 1993.

Menezes, Alfred, Van Orschott, Paul, and Vanstone, Scott. Handbook of Applied Cryptography. CRC Press, 1997.

IEEE Draft Standard P1363: Standards for Public Key Cryptography, available at

<http://stdsbbc.ieee.org/groups/1363/index.html>

Certicom ECC Tutorials and Whitepapers found at:

<http://www2.certicom.com/ecc/index.htm>