Global Information Assurance Certification Paper

Elliptic Curve Cryptosystems – an overview

Leonard Jacobs

March 24, 2001

It is assume that readers of this paper possess a basic understanding of algebraic/geometric mathematics and cryptographic methodologies. You will find a glossary, at the end of this paper, to assist you in better understanding the terminology used.

In 1985, Neil Koblitz from the University of Washington and Victor Miller, who was working at IBM at that time, independently proposed the Elliptic Curve Cryptosystem (ECC), whose security rests on the discrete logarithm problem over the points on an elliptic curve. ECC can be used to provide both a digital signature scheme and an encryption scheme. ECC represents an alternative to older forms of public-key cryptography and offers certain advantages; which will be explored later in this paper. [1]

Elliptic Curves [1, 2]

To understand what ECC entails, one must understand the arithmetic involved with elliptic curves. Elliptic curves as algebraic/geometric entities have been studied extensively for the past 150 years.

An elliptic curve consists of elements (x, y) satisfying the equation

$$y^2 = x^3 + ax + b \pmod{p}$$

for two numbers *a* and *b*. If (*x*, *y*) satisfies the above equation then *P*=(*x*, *y*) is a point on the elliptic curve. The elliptic curve formula is slightly different for some fields.

Elliptic curves can be defined over any field such as real, fractional, and complex. Elliptic curves used in cryptosystems are typically defined over finite fields (integer modulo a prime number).

A finite field consists of a finite set of elements together with two operations, addition and multiplication, that satisfy certain arithmetic properties. Finite fields often used in cryptography are F_p; where p is a prime number, and the field F_2m.

Using some particularly profound mathematics, it is possible to define the addition of two points on the elliptic curve. Suppose *P* and *Q* are both points on the curve, then

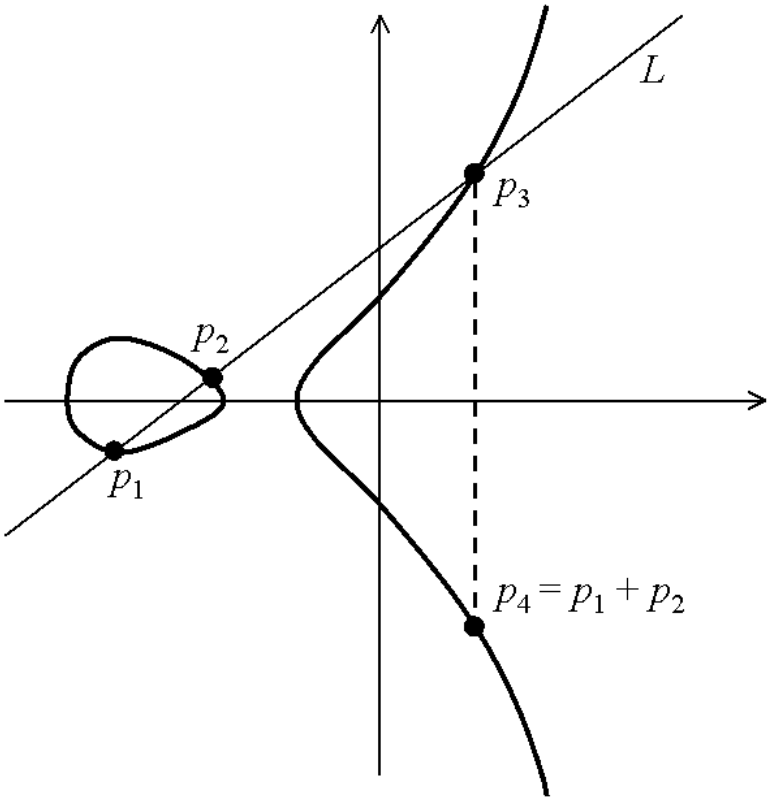
$$P + Q$$

will always be another point on the curve.

All public-key cryptosystems exploit the mathematical properties of large finite groups Two groups used in cryptography are Z_n, the additive group of integers modulo a number *n*; and Z_p^*, the multiplicative group of integers modulo a prime number *p*.

Example of an Elliptic Curve

Figure 1: [2]



The set of points on an elliptic curve forms a group under addition, where addition of two points on an elliptic curve is defined according to a set of simple rules. For example, consider the two points p_1 and p_2 in Figure 1. Point p_1 plus point p_2 is equal to point $p_4 = (x, -y)$, where $(x, y) = p_3$ is the third point on the intersection of the elliptic curve and the line L through p_1 and p_2.

Elliptic Curve Discrete Logarithm Problem (ECDLP) [1, 2]
The elliptic curve discrete logarithm problem (ECDLP) can be stated as follows. Fix a prime p and an elliptic curve. xP represents the point P added to itself x times. Suppose Q is a multiple of P, so that

$$Q = xP$$

for some x. Then the ECDLP is to determine x given P and Q.

The general conclusion of leading cryptographers is that the ECDLP requires fully exponential time to solve. The security of ECC is dependent on the difficulty of solving the ECDLP.

Security of ECC
Mathematicians have given considerable attention to ECDLP. Like the other types of cryptographic problems, no efficient algorithm is known to solve the ECDLP. The ECDLP seems to be particularly harder to solve. Moderate security can be achieved with the ECC using an elliptic curve defined modulo a prime *p* that is several times shorter than 230 decimal digits.

An elliptic curve cryptosystem implemented over a 160-bit field currently offers roughly the same resistance to attack, as would a 1024-bit RSA modulus. Moderate security can be achieved with the ECC using an elliptic curve defined modulo a prime *p* that is several times shorter than 230 decimal digits.

At the security level of 10^{20} MIPS years, it takes a 300-bit key in ECC to equal the strength of a 2048 bit key in either RSA or DSA. The currently acceptable security level is 10^{12} MIPS years. The security gap between the systems grows as the key size increases.

In general, no major weaknesses with ECC have been discovered. However, it has been one reported that a 108-bit elliptic curve encryption key was cracked in July 2000. It took 9,500 computers running in parallel for four months, connected via the Internet. It would take 500 years of processing on a single 450 MHz personal computer to perform the same key cracking. [3]

There have been weak classes of elliptic curves identified such as supersingular elliptic curves and some anomalous elliptic curves. Implementations, such as ECDSA, merely check for weaknesses and eliminate any possibility of using these "weak" curves. [4]

Elliptic Curve Digital Signature Algorithm (ECDSA) [5, 6]
A major application of ECC is ECDSA. ECC applications work extremely well with small amounts of data such as digital signatures. The ECDSA is the elliptic curve analogue of the Digital Signature Algorithm (DSA).

The key generation, signature generation, and signature verification procedures for ECDSA are as follows:

ECDSA Key Generation -
1. Entity A selects an elliptic curve E defined over Z_p. The number of points in $E(Z_p)$ should be divisible by a large prime n.
2. Select a point P • $E(Z_p)$ of order n.
3. Select a statistically unique and unpredictable integer d in the interval [1, n-1].
4. Compute Q = dP.
5. A's public key is (E, P, n, Q). A's private key is d.

ECDSA signature generation -
1. Entity a selects a statistically unique and unpredictable integer k in the interval [1, n-1].

2. Compute $kP = (x_1, y_1)$ and $r = x_1 \bmod n$. To avoid a security condition, x_1 should not equal 0.
3. Compute $k^{-1} \bmod n$.
4. Compute $s = k^{-1} \{h(m) + dr\} \bmod n$. h is the Secure Hash Algorithm (SHA-1).
5. If $s = 0$, then go to Step 1. If $s = 0$, then $s^{-1} \bmod n$ does not exist and s^{-1} is required in the signature verification process.
6. The signature for the message m is the pair of integers (r, s).

ECDSA signature verification –

1. Entity B obtains an authentic copy of Entity A's public key (E, P, n, Q).
2. Verify that r and s are integers in the interval [1, n-1].
3. Compute $w = s^{-1} \bmod n$ and h(m).
4. Compute $u_1 = h(m)w \bmod n$ and $u_2 = rw \bmod n$.
5. Compute $u_1P + u_2Q = (x_0, y_0)$ and $v = x_0 \bmod n$.
6. Entity B accepts the signature if and only if v =r.

Instead of each entity choosing to generate its own elliptic curve, the entities can use the same curve E over Z_p and point P of order n. In this situation, an entity's public key consists of just point Q. This results in smaller public key sizes.

ECDSA defines private keys and per-signature values as statistically unique and unpredictable rather than merely random as defined in the DSA standard.

ECDSA utilizes a method called point compression, which allows a point on an elliptic curve to be represented, by one field element and one additional bit. This leads to substantially reduced sizing in public-key certificates. Reductions are typically 25 percent compared to asymmetric algorithms.

The primality test in DSA is probabilistic. ECDSA provides for deterministic primality testing. This allows a high security application to verify that claimed prime numbers are indeed prime numbers.

In ECDSA, 100 percent of signatures generated will verify due to mandatory analogous bounds checking. DSA has optional bounds checking on signature generation.

In ECDSA, authorized use of a private key has been made explicit. DSA states that the integrity of signed data is dependent upon the prevention of unauthorized disclosure, modification, substitution, insertion, and deletion of the private key value or per-signature value. Unlike ECDSA, unauthorized use is not explicitly prohibited in DSA.

Advantages of ECC [4]
ECC leads to more efficient implementations than other public-key systems due to its extra strength provided by the difficulty to solve the ECDLP.

The biggest advantage of ECC is key size. For example, a typical key size for the RSA algorithm is 1024 bits; which would take approximately 10^{11} MIPS years to break. In comparison, an ECC key size is 160 bit offers the same level of security.

Computational efficiencies are achieved with ECC. ECC does not require processing of prime numbers to achieve encryption unlike other public-key cryptosystems. ECC is roughly 10 times faster than either RSA or DSA.

ECC offers considerable bandwidth savings over the other types of public-key cryptosystems when being used to transform short messages such as the typical implementation of ECDSA. Bandwidth savings is about the same as other public-key cryptosystems when transforming long messages.

These advantages lead to higher speeds, lower power consumption, and code size reductions. Implementations of ECC are particularly beneficial in applications where bandwidth, processing capacity, power availability, or storage is constrained. Such applications include wireless transactions, handheld computing, broadcast, and smart card applications.

Review

Koblitz and Miller first proposed ECC in 1985. ECC is based on the properties of elliptic curves. Elliptic curves are defined by $y^2 = x^3 + ax + b \pmod{p}$. The security provided by ECC is stronger than other public-key cryptosystems due to the characteristics of ECDLP. ECC typically requires a smaller size key than other public-key encryption methods for the same or better security. ECC can be used for digital signatures and encryption. The ECDSA standard provides for better security than other forms of public-key encryption and signature generation. Applications requiring low power, low memory, and less bandwidth are ideal with ECC.

Glossary [7]

algorithm – A series of steps used to complete a task.

ciphertext – Encrypted data.

cryptosystem – An encryption, decryption algorithm, together with all possible plaintexts, ciphertexts, and keys.

decryption – The reverse of encryption.

digital signature – The encryption of a message with a private key.

discrete logarithm – Given two elements d, g, in a group such that there is an integer r satisfying $g^r = d$.

discrete logarithm problem – The problem of given d and g in a group, to find r such that $g^r = d$. For some groups, this problem is a hard problem that can be used in public-key cryptography.

encryption – The transformation of plaintext into an apparently less readable form through a mathematical process.

field – A mathematical structure with multiplication and addition that behave as they do with real numbers.

group – A mathematical structure in which elements are combined.

hard problem – A computationally intensive problem that is difficult to solve. The basis of most cryptosystems.

hash function – A function that takes a variable size input and has a fixed size output.

key – A string of bits that allow people to encrypt and decrypt data by determining the mapping of plaintext to cipher text.

MIPS – Million instructions per second. A measure of computational speed in computers.

MIPS year - A MIPS year represents a computing time of one year on a machine capable of performing one million instructions per second.

plaintext – The data to be encrypted.

prime number – Any integer greater than 1 that is divisible by 1 and itself.

private key – This key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures.

public key – This is the key made public to all. It is primarily used for encryption but is also used to verify digital signatures.

public-key cryptography – Cryptography based on methods using a public key and a private key.

RSA algorithm – A public-key cryptosystem based on the factoring problem. RSA stands for Rivest, Shamir, and Adleman, the developers of the RSA public-key cryptosystem.

smart card – A card, not much bigger than a credit card, that contains a computer chip and is used to store or process information.

References and Bibliography

1. "Certicom – Online Tutorial", http://www.certicom.ca/research/ec20.html

2. "What are Elliptic Curves?" http://www.rsasecurity.com/rsalabs/faq/2-3-10.html

3. "108-Bit Elliptic-Curve Encryption Key Cracked", http://www.nikkeibp.asiabiztech.com/nea/200007/late_106440.html

4. "Current Public-Key Cryptographic Systems", http://www.certicom.ca/

5. "Elliptic Curve DSA (ECDSA): An Enhanced DSA", http://www.certicom.ca/research/wecdsa.html

6. "The Elliptic Curve Digital Signature Algorithm (ECDSA); Don Johnson, Alfred Menezes, Scott Vanstone; http://www.certicom.ca/

7. Glossary, http://www.certicom.ca/research/glossary.html

Last Updated: May 28th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event