Interested in learning more about security?

SANS Institute
InfoSec Reading Room

This paper is from the SANS Institute Reading Room site. Reposting is not permitted without express written permission.

# Elliptic Curve Cryptography and Smart Cards

Elliptic curve cryptosystems (ECCs) are becoming more popular because of the reduced number of key bits required in comparison to other cryptosystems (for example, a 160 bit ECC has roughly the same security strength as 1024 bit RSA). In addition, ECC satisfies smart cards requirements in terms of memory, processing and cost. In this report, I will present a background on ECC including the basics and some ECC techniques. Then, I will talk about smart cards, their constraints and ECC implementation options.

Copyright SANS Institute
Author Retains Full Rights

AD

# Elliptic Curve Cryptography and Smart Cards

AHMAD KHALED M. AL-KAYALI

17 February, 2004
GIAC Security Essentials Certification (GSEC)
Practical Assignment Version 1.4b, Option 1

© SANS Institute 2004, Author retains full rights.

**ABSTRACT**

Elliptic curve cryptosystems (ECCs) are becoming more popular because of the reduced number of key bits required in comparison to other cryptosystems (for example, a 160 bit ECC has roughly the same security strength as 1024 bit RSA). In addition, ECC satisfies smart cards requirements in terms of memory, processing and cost. In this report, I will present a background on ECC including the basics and some ECC techniques. Then, I will talk about smart cards, their constraints and ECC implementation options.

© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

# TABLE OF CONTENTS

**TABLE OF CONTENTS** ..... **2**

**1. INTRODUCTION** ..... **3**

**2. SOME CRYPTOGRAPHY CONCEPTS** ..... **3**

*Cryptographic Goals* ..... 3
*Types of Cryptography* ..... 3

**3. ELLIPTIC CURVE THEORY** ..... **4**

*Elliptic Curves over Finite Fields* ..... 4
*Elliptic Curves over GF(p)* ..... 5
*Elliptic Curves over* $GF(2^K)$ ..... 6

**4. ELLIPTEC CURVE CRYPTOGRAPHY** ..... **7**

*Elliptic Curve Diffie-Hellman Key Exchange* ..... 7
*Elliptic Curve Encryption/Decryption* ..... 8
*Elliptic Curve Digital Signature Algorithm (ECDSA)* ..... 8

**5. SMART CARDS AND ECC** ..... **9**

*Smart Card Types* ..... 10
*How ECC Fits Smart Cards* ..... 10
*ECC Implementation Choices for Smart Cards* ..... 11

**6. CONCLUSION** ..... **12**

**REFERENCES** ..... **13**

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

# 1. INTRODUCTION

In 1985 Niel Koblitz and Victor Miller proposed the Elliptic Curve Cryptosystem (ECC) [8]. ECC is basically a method based on the Discrete Logarithm Problem over the points on an elliptic curve. Since that time, ECC has received considerable attention from mathematicians around the world, and no significant weaknesses in the algorithm have been demonstrated. Although there are still some doubts in the reliability of this method, several encryption techniques have been developed recently using these properties.

Whenever the cryptographic problem appears so difficult to crack or break, it means that key sizes can be reduced in size considerably especially when compared to the key size used by other cryptosystems [8]. This made ECC a challenge to the RSA, one of the most popular public-key methods known. ECC is showing to offer equivalent security to RSA but with much smaller key size.

ECC not only permits the reduction of the key size and, but also ECC is able to do operations very fast. In addition, processing power can be reduced in ECC. All those features allow ECC to be a convenient environment for smart cards.

# 2. SOME CRYPTOGRAPHY CONCEPTS

## *Cryptographic Goals*

Generally, a good cryptography scheme must satisfy a combination of four different goals [2]:

- Authentication: Allowing the recipient of information to determine its origin, that is, to confirm the sender's identity. This can be done through something you know or you have. Typically provided by digital signature.
- Nonrepudiation: Ensuring that a party to a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated. Typically provided by digital signature.
- Data integrity: A condition in which data has not been altered or destroyed in an unauthorized manner. Typically provided by digital signature.
- Confidentiality: Keeping the data involved in an electronic transaction private. Typically provided by encryption.

## *Types of Cryptography*

There are two main types of cryptography. Those are public-key and symmetric-key. Public-key is a form of cryptography in which two digital keys are generated, one is private, which must not be known to another user, and one is public, which may be made available in public. These keys are used for either encrypting or signing messages. The public-key is used to encrypt a message and the private-key is used to decrypt the message. However, in another scenario, the private-key is used to sign a


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

message and the public-key is used to verify the signature. The two keys are related by a hard one-way (irreversible) function, so it is computationally infeasible to determine the private key from the public key. Since the security of the private key is critical to the security of the cryptosystem, it is very important to keep the private key secret. This public-key system has the problem of being slow. On the other hand, the system has powerful key management and, even more importantly, public-key cryptography has the ability to implement digital signatures in an efficient way.

However, symmetric-key is a form of cryptography in which two parties that want to communicate can share a common and secret key. Each party must trust the other not to tell the common key to anyone else. This system has the advantage of encrypting large amount of data efficiently. However, the problem rises when it comes to key management over large number of users. [1]

## 3. ELLIPTIC CURVE THEORY

Elliptic curves are known so because they are described by cubic equations, similar to those used in ellipsis calculations. The general form for elliptic curve equation is [6]:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

Figure 1 shows a graph drawn by a tool[1] available at Certicom (www.certicom.com) of some elliptic curve.

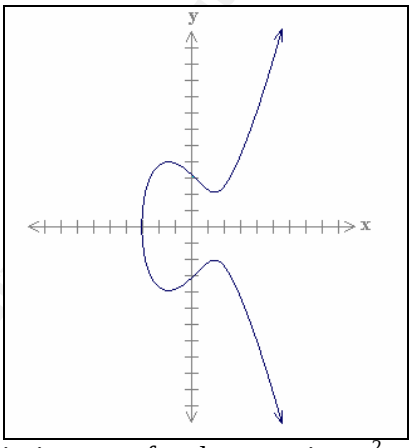


Figure 1: Elliptic curve for the equation $y^2 = x^3 - 6x + 10$

### *Elliptic Curves over Finite Fields*

A finite field is a set of elements that have a finite order (number of elements). The order of *Galois Field* (*GF*) is normally a prime number or a power of a prime number. There are many ways of representing the elements of the finite field. Some representations may lead to more efficient implementations of the field arithmetic in hardware or in software. The elliptic curve arithmetic is more or less complex depending on the finite field where the elliptic curve is applied. $GF(2^k)$ and $GF(p)$ are considered in this report

[1] http://www.certicom.com/content/live/resources/ecc_tutorial/ecc_javaCurve.html


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

because of their popularity in ECC. In fact, there are no security or standardization differences exist between the two types. However, performance and cost differences can arise when a smart card application is to be implemented as will be shown later in this report. [1,10]

### *Elliptic Curves over GF(p)*

An elliptic curve over *GF(p)* , where p is a prime, can be defined as the points *(x, y)* satisfying the elliptic curve equation:

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

Where $4a^3 + 27b^2 \neq 0 \pmod p$ and $x, y, a, b$ Î $GF(p)$.

In addition to the points satisfying the curve equation *E*, a point at infinity, j , is also defined. With a suitable definition of addition and doubling of points, this enables the points of an elliptic curve to form a group with addition and doubling of points being the group operation, and the point at infinity being the identity element.

In order to achieve an efficient implementation of elliptic curves, firstly field arithmetic (modular addition, subtraction, multiplication and inversion) must be available. These operations are then used in the algorithms for addition and doubling of points.

The addition of two different points on the elliptic curve is computed as shown below [9]:

$$(x_1 , y_1) + (x_2 , y_2) = (x_3 , y_3) \text{ ; where } x_1 \text{ ¹ } x_2$$

$$\text{l} = (y_2 - y_1)/(x_2 - x_1)$$

$$x_3 = \text{l}^2 - x_1 - x_2$$

$$y_3 = \text{l}(x_1 - x_3) - y_1$$

Figure 2 shows how two points can be added graphically on the elliptic curve. Assume that P and Q are two distinct points on the elliptic curve *E*. Since we are intersecting a line with a cubic curve, the straight line joining *P* and *Q* must intersect through a third point on the curve; we will call it *-R*. If we reflect the point *–R* in the x-axis, we will get another point called *R*, where *R = P+Q*.

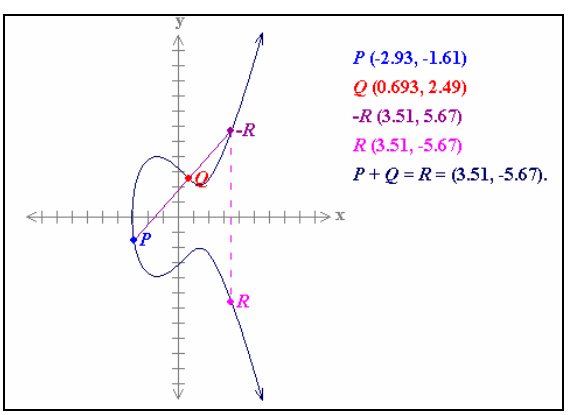


Figure 2: Two-point addition (R=P+Q) operation over elliptic curve


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

The addition of a point to itself (doubling a point) on the elliptic curve is computed as shown below [9]:

$$(x_1 , y_1) + (x_1 , y_1) = (x_3 , y_3) \text{ ; where } x_1 \neq 0$$
$$l = (3(x_1)^2 + a) /(2y_1)$$
$$x_3 = l^2 - 2x_1$$
$$y_3 = l(x_1 - x_3) - y_1$$

Figure 3 shows how a point can be doubled graphically on the elliptic curve. Assume we want to double a point *P* on the elliptic curve. We take the tangent line to the curve and passing by *P*. The line must intersect the curve through another point; we will call it *-R*. Again, we reflect the point *–R* in the x-axis to the point *R* where *R=2P*.

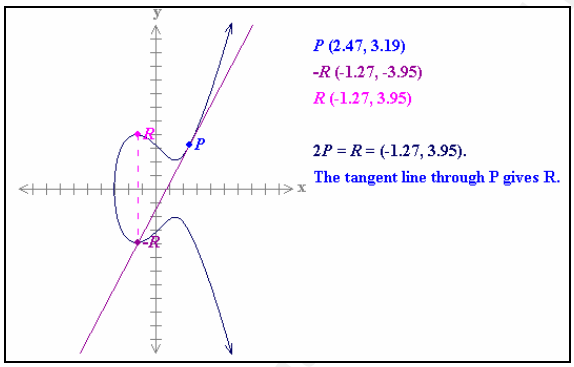


Figure 3: Point doubling (R=2P) operation over elliptic curve

## *Elliptic Curves over GF($2^k$)*

*GF($2^k$)* is called a characteristic two field or a binary finite field. It can be viewed as a vector space of dimension *k* over the field *GF(2)* that consists of the elements 0 and 1. To clarify, there exist *k* elements $(x_0 , x_1 , x_2 . . . , x_{k-1})$ in *GF($2^k$)* such that each element $x \in GF(2^k)$ can be uniquely written in the form:

$$x = a_0 x_0 + a_1 x_1 . . . + a_{k-1} x_{k-1}$$

Where $a_i \in GF(2)$.

Such a set $\{x_0 , x_1 , x_2 . . . , x_{k-1}\}$ is called a basis of *GF($2^k$)* over *GF(2)*. Given such a basis, a field element *x* can be represented as the bit string $(a_0 a_1 . . . a_{k-1})$. Performing addition of any field elements can be achieved easily by bit-wise XOR-ing the vector representations of the elements. The multiplication rule depends on the selected basis. *GF($2^k$)* over *GF(2)* has many different bases. Some bases lead to more efficient implementations of the arithmetic in *GF($2^k$)* than other bases. The most popular two bases used are the polynomial and the normal bases. Since elements in one basis representation can be efficiently converted to elements in the other basis representation by using an appropriate change-of-basis matrix, interoperability between systems using the two different types of field representation can be achieved easily. The elliptic curve equation over *GF($2^k$)* is:

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

Where x, y, a, b $\in$ *GF($2^k$)* and b $\neq$ 0.


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

The addition of two different points on the elliptic curve is computed as shown below [9]:

$$(x_1 , y_1) + (x_2 , y_2) = (x_3 , y_3) \text{ ; where } x_1 \neq x_2$$
$$l = (y_2 + y_1)/(x_2 + x_1)$$
$$x_3 = l^2 + l + x_1 + x_2 + a$$
$$y_3 = l(x_1 + x_3) + x_3 + y_1$$

The addition of a point to itself (Doubling a point) on the elliptic curve is computed as shown below [9]:

$$(x_1 , y_1) + (x_1 , y_1) = (x_3 , y_3) \text{ ; where } x_1 \neq 0$$
$$l = x_1 + (y_1)/(x_1)$$
$$x_3 = l^2 + l + a$$
$$y_3 = (x_1)^2 + (l + 1)\, x_3$$

## 4. ELLIPTEC CURVE CRYPTOGRAPHY

The point addition in elliptic curves is the basic operation to make it used in cryptography. Calculating the point *k(x, y)* from *(x, y)* is feasible to figure. *k(x, y)* can be computed by repeated point additions such as [8]:

$$\underbrace{(x,y)+(x,y)+\ldots+(x,y)}_{k\ times} = k \times (x,y)$$

Where $k \in N$ and *(x, y)* is a point on the elliptic curve.

However, it is very hard to determine the value of *k* knowing the two points: *k(x, y)* and *(x, y)*. This leads to the definition of Elliptic Curve Logarithm Problem (ECDLP), which is defined as: Let *E* be an elliptic curve defined over a finite field, and let *P* be a point (called base point) on *E* of order *n*. Given *Q* as another point on *E*, the Elliptic Curve Discrete Logarithm Problem (ECDLP) is to find the integer $l, 0 \le l \le n-1$, such that:

$$Q = lP.$$

This property leads to several algorithms for cryptography. Some of these techniques will be introduced in the following subsections.

### *Elliptic Curve Diffie-Hellman Key Exchange*

Symmetric-key (also known as secret-key) cryptosystems are normally used for encryption/decryption purposes, because it is faster than public-key cryptosystems. Symmetric-key cryptosystems requires a secret key to be agreed upon before the cryptographic process starts. This agreement can be performed by Diffie-Hellman key exchange technique on which elliptic curve idea can be applied. The elliptic curve Diffie-Hellman key exchange method is described by the following example.

Suppose that users A and B want to agree upon a secret key, which will be used for secret key cryptography. Users A and B choose a finite field, *GF(p)* for example, and an elliptic curve *E* is defined over this field. They also construct a randomly chosen point


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

*(x, y)* lying on the elliptic curve *E*. We will refer to *(x, y)* as the base point of the cryptosystem. The finite field, the elliptic curve, and the base point are all to be known publicly.

User A then randomly chooses a large integer *a* Î *GF(p)* and keeps *a* secret. User A now computes the point *a(x, y)* which will lie on *E*. User B does the same: B randomly chooses a large integer *b* and computes *b(x, y)*. Both A and B make *a(x, y)* and *b(x, y)* publicly known [8]. In other words, these act as public keys. The secret key that A and B use to encrypt messages sent to each other is *ab(x, y)*•, which both A and B can compute. User A knows *a* and *b(x ,y)*, and so can find *ab(x, y).* Whereas, B knows *b* and *a(x, y)*, and so can find *ab(x, y)*. The security of this system lies in the fact that a third party C, for example, knows only *a(x, y)* and *b(x, y)*, and unless C can solve the elliptic curve discrete logarithm problem there is no efficient way to break the encryption.

## *Elliptic Curve Encryption/Decryption*

There are many ways to apply elliptic curves for encryption/decryption purposes. A simple method will be introduced here to give the flavor of elliptic curve encryption/decryption techniques. Assume working with *GF(p)* finite field and an elliptic curve *E*. The users randomly chose a *base point (x, y)*, lying on the elliptic curve *E*. The plaintext (the original message to be encrypted) is coded into an elliptic curve point $(x_m, y_m)$. Each user selects a private key *n* (which can be placed in his smart card) and compute his public key $P = n(x, y)$. For example, user A's private key is $n_A$ and his public key is $P_A = n_A(x, y)$.

For any one to encrypt and send the message point $(x_m, y_m)$ to user A, he/she needs to choose a random integer *k* and generate the ciphertext $C_m = \{k(x, y) , (x_m, y_m)+ kP_A \}$. The ciphertext pair of points uses A's public key, where only user A can decrypt the plaintext using his private key [8].

To decrypt the ciphertext $C_m$, the first point in the pair of $C_m$, $k(x, y)$, is multiplied by A's private key to get the point: $n_A (k(x, y))$. Then this point is subtracted from the second point of $C_m$, the result will be the plaintext point $(x_m, y_m)$. The complete decryption operations are [8]:

$$((x_m, y_m)+ kP_A) - n_A (k(x, y)) = (x_m, y_m)+ k (n_A(x, y)) - n_A (k(x, y)) = (x_m, y_m)$$

## *Elliptic Curve Digital Signature Algorithm (ECDSA)*

Elliptic Curve Digital Signature Algorithm (ECDSA) is elliptic curve analogue of the Digital Signature Algorithm (DSA). The signature has the property that it can be produced by only one single individual (or smart card) who has the private key, but can be verified by anyone who receives the message. The process of ECDSA is composed of three main steps: key generation, signature generation and signature verification. Each step is described as follows [5]:


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

*ECDSA key generation*
Each user of the scheme does the following:

1. Select an elliptic curve $E$ over a finite field, say $GF(p)$. The number of points on $E$ should be divisible by a large prime $n$.
2. Select a point $P = (x,y) \in GF(p)$ of order $n$.
3. Select an unpredictable integer $d$ in the range $[1, n-1]$. $d$ will act as the private key
4. Compute $Q=dP$
5. The user's public key is $(E, P, n, Q)$.

*ECDSA signature generation*
To sign a message, $m$, the user does the following:

1. Select a random integer $k$ in the range $[1, n-1]$.
2. Compute $(x_1, y_1) = kP = k(x,y)$, and set $r = x_1 \bmod n$. If $r$ is zero then go back to step 1. In other words, if $r=0$ then the signing equation $[s = k^{-1}(h(m) + dr) \bmod n]$ will not involve the private key $d$.
3. Compute $k^{-1} \bmod n$.
4. Compute $s = k^{-1}(h(m) + dr) \bmod n$, where $h$ is the hash value obtained from a suitable hash algorithm (for example, the Secure Hash Algorithm, SHA-1).
5. If $s=0$ go to step 1. This because if $s$ is zero then $s^{-1} \bmod n$ does not exist and $s^{-1} \bmod n$ is needed in the signature verification.
6. The signature to be included in the message $m$ is the pair of integers $(r, s)$.

*ECDSA signature verification*
To verify the signature $(r, s)$ on the message $m$, the following should be done:

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

## 5. SMART CARDS AND ECC

A smart card is a plastic card about the size of a credit card, with an embedded microchip that can be loaded with data, used for telephone calling, electronic cash payments, health care, identification and other applications [9]. One can see that many of those applications are the right place to use encryption and/or digital signature. In addition, for smart cards to be used practically, they need to be inexpensive.

Smart cards have the fact that their stored data can be protected against unauthorized access and tampering. Therefore, smart cards can safely contain sensitive data. Example of sensitive data is the private key which is used to perform signature or decryption. The private key can be protected by the smart card since it never leaves the smart card. Smart card is considered to be ideal cryptographic token.


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

## *Smart Card Types*

*Memory Cards*

Though referred to as smart cards, memory cards are typically much less expensive and much less functional than microprocessor cards. They contain EEPROM and ROM memory, as well as some security logic. In the simplest case, security logic exists to prevent writing and erasing of the data. More complex cases allow for memory read access to be restricted by encryption. Typical memory card applications are pre-paid telephone cards and health insurance cards.

*Microprocessor Cards*

Components of this type of architecture include a CPU, RAM, ROM and EEPROM. Majority of smart cards in the market have between 128 and 1024 bytes of RAM, between 1 and 16 kilobytes of EEPROM, between 6 and 16 kilobytes of ROM and an 8-bit CPU clocked at a 3.57 megahertz. The operating system is typically stored in ROM, the CPU uses RAM as its working memory and most of the data is stored in EEPROM. Usually, RAM requires four times as much space as EEPROM, which in turn requires four times as much space as ROM.

*Cryptographic Coprocessor Cards*

Although technically these are in the category of microprocessor cards, they are separated here because of differences in cost and functionality. Because some cryptographic algorithms require very large calculations, an 8-bit microprocessor with very little RAM can take on the order of several minutes to perform operation over large private keys. However, if a cryptographic coprocessor (a dedicated hardware component for cryptographic processing) is added to the architecture, the time required for this same operation is reduced to around a few hundred microseconds. However. there is a drawback which is the cost. The addition of a cryptographic coprocessor can increase the cost of today's smart cards by 20% to 30%[1].

*Contactless Smart Cards*

Though the reliability of smart card contacts has improved to very acceptable levels over the years, contacts are one of the most frequent failure points any electromechanical system due to dirt, wear… etc. The contactless card solves this problem and also provides the issuer an a high degree freedom during use. Cards need no longer be inserted into a reader, which could improve end user acceptance. No chip contacts are visible on the surface of the card. However, despite these benefits, contactless cards have not yet seen wide acceptance. The cost is higher and not enough experience has been gained to make the technology reliable enough. [8]

## *How ECC Fits Smart Cards*

*Less Memory and Shorter Transmission Times*

ECDLP algorithm leads to a very strong security with relatively small keys. When the key becomes smaller, the memory needed to store keys is smaller and, consequently, the less data needs to transfer between the card and the application. Therefore, the transmission time is shorter.


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

*Scalability*
Smart card applications always require stronger security that can be achieved with longer keys. However, ECC can provide the security with relatively fewer extra system resources. This means that with ECC, smart cards can keep their cost and provide a higher level of security at the same time.

*No Coprocessor*
ECC reduces the processing times very much because of the nature of actual computations (especially in case of $GF(2^k)$ where there are no modular operations). Other systems have a dedicated crypto coprocessor to do the computations. Coprocessor has the problem of increasing both the area and the cost. However, in the case of ECC, the algorithm can be implemented in the available CPU with no additional hardware.

*On Card Key Generation*
For true nonrepudiation, the private key must be kept secret and inaccessible to anyone. In current public-key systems, cards are personalized (keys are loaded or injected into the cards) in a secure environment to meet this security issue. Because of the complexity of the computation required, generating keys on the card is inefficient and typically impractical. With ECC, the time needed to generate a key is so short and can be done with the limited computing power of a smart card, provided a good random number generator is available. This means that the card personalization process can be more efficient for applications in which nonrepudiation is important.

## *ECC Implementation Choices for Smart Cards*

In general, the use of $GF(2^k)$ offers significant performance advantages over $GF(p)$. The reason is for that is the existence of modular operations in case of $GF(p)$ [4].This is true for a low-cost 8-bit smart card which has no crypto coprocessor. To achieve closely equivalent performance with $GF(p)$, a crypto coprocessor is required. This additional crypto coprocessor increases the cost of each chip by 20% to 30%, which adds three to five dollars to the cost of the card. With $GF(2^k)$, a smart card is less expensive because a coprocessor is not needed.

In environments in which an arithmetic processor is already available, the performance of $GF(p)$ can be improved so that in some cases it exceeds the performance of $GF(2^k)$. This is true for platforms with crypto coprocessor such as some types of smart cards. If a crypto coprocessor is already available on the smart card or the cost is not a big deal, then $GF(p)$ would offer performance advantages over $GF(2^k)$ implemented without a dedicated hardware component.

Additionally, point compression allows the points on an elliptic curve to be represented with fewer bits of data. In smart card implementations, point compression is essential because it reduces not only the storage space for keys on the card, but also the amount of data that needs to be transmitted to and from the card. It can be accomplished with


© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

negligible computation using $GF(2^k)$, but can affect $GF(p)$ implementations considerably. $GF(2^k)$ hardware implementations offer significant performance and area size advantages over $GF(p)$ hardware implementations. Smart cards requiring a number of different cryptographic services with extremely fast performance would require cryptographic coprocessors. Existing crypto coprocessors, which are optimized for modular arithmetic needed on $GF(p)$, do not substantially increase the performance of $GF(2^k)$ arithmetic. A coprocessor designed specifically to optimize $GF(2^k)$ would take up less space on the smart card and cost, and would provide superior performance to an $GF(p)$ implementation.

## 6. CONCLUSION

In this paper, I demonstrated the Elliptic Curve Cryptography (ECC). The demonstration included some of the theoretical and practical aspects of ECC. Most of the studies currently made have shown that ECC is the most convenient cryptosystem for the smart cards. Saving time, cost and area are the reasons behind this for smart cards. On the other hand, the fact that the elliptic curve cryptosystem implementation is much more complicated and requires deeper mathematical understanding than the other cryptography implementation (for example RSA), makes it more susceptible to errors. Certainly, ECC systems solved some major problems which exist in others. Clearly time will tell whether such a strong system is here to stay.

© SANS Institute 2004, Author retains full rights.



© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

## REFERENCES


1) Certicom Corp. "The Elliptic Curve Cryptosystem for Smart Cards." May 1998. URL: http://www.comms.engg.susx.ac.uk/fft/crypto/ECC_SC.pdf (15 Jan. 2004)
2) Cole, Eric, Jason Fossen, Stephen Northcutt, Hal Pomeranz. SANS Security Essentials with CISSP CBK, Version 2.1.USA: SANS Press, 2003.
3) Crutchley, Duncan Alexander. "Cryptography And Elliptic Curves." May 1999. URL: http://www.dacrutchley.plus.com/files/ecc_project.zip (2 Jan. 2004)
4) Hitchcock, Y., Edward Dawson, Andrew Clarck, Paul Montague. "Implementing an efficient elliptic curve cryptosystem over GF(p) on a smart card." 24 October 2002. URL: http://anziamj.austms.org.au/V44/CTAC2001/Hitc/Hitc.pdf (2 Feb. 2004)
5) Jonson, D., Alfred Menezes. "Ellipic Curve DSA (ECDSA): An Enhanced DSA." 24 February 2000. URL: http://citeseer.nj.nec.com/cache/papers/cs/8755/http:zSzzSzcacr.math.uwaterloo.cazSz~ajmenezezSzpublicationszSzecdsa.pdf/johnson99elliptic.pdf (8 Feb. 2004)
6) Pietiläinen , Henna. "Elliptic curve cryptography on smart cards." October 2000. URL: http://henna.laurikari.net/Dippa/di.pdf (2 Feb. 2004)
7) Ranket, Wolfgang, Wolfgang Effing. Smart Card Handbook, Second Edition. England: Willey, 2000:18-26.
8) Stallings, William. Cryptography and Network Security: Principles and Practice, Second Edition. New Jersey: Prentice Hall Inc., 1999.
9) Von York, E. "Elliptic Curves Over Finite Fields." 7 May 1992. URL: http://www.mapleapps.com/categories/mathematics/algebra/code/elliptic/elliptic.pdf (24 Jan. 2004)
10) Woodbury, Adam D., Daniel V. Bailey, Christof Paar. "Elliptic Curve Cryptography on Smart Cards without Coprocessors." The Fourth Smart Card Research and Advanced Applications (CARDIS 2000) Conference. 22 September 2000. URL: http://www.crypto.ruhr-uni-bochum.de/Publikationen/texte/WoodburyBaileyPaarCARDIS.pdf (24 Jan. 2004)




© SANS Institute 2004, As part of the Information Security Reading Room Author retains full rights.

Last Updated: May 22nd, 2013

# Upcoming SANS Training

**Click Here for a full list of all Upcoming SANS Events by Location**

| | | | |
|---|---|---|---|
| **SANS Malaysia @ MCMC 2013** | **Cyberjaya, MY** | **Jun 03, 2013 - Jun 08, 2013** | **Live Event** |
| **SANS Pen Test Berlin 2013** | **Berlin, DE** | **Jun 03, 2013 - Jun 08, 2013** | **Live Event** |
| **Industial Control Systems Security Training - Houston** | **Houston, TXUS** | **Jun 10, 2013 - Jun 15, 2013** | **Live Event** |
| **Security Impact of IPv6 Summit 2013** | **Washington, DCUS** | **Jun 14, 2013 - Jun 16, 2013** | **Live Event** |
| **SANSFIRE 2013** | **Washington, DCUS** | **Jun 14, 2013 - Jun 22, 2013** | **Live Event** |
| **SANS Canberra 2013** | **Canberra, AU** | **Jul 01, 2013 - Jul 13, 2013** | **Live Event** |
| **Digital Forensics & Incident Response Summit 2013** | **Austin, TXUS** | **Jul 09, 2013 - Jul 16, 2013** | **Live Event** |
| **SANS London Summer 2013** | **London, GB** | **Jul 09, 2013 - Jul 16, 2013** | **Live Event** |
| **SANS Rocky Mountain 2013** | **Denver, COUS** | **Jul 14, 2013 - Jul 20, 2013** | **Live Event** |
| **SANS Mumbai 2013** | **Mumbai, IN** | **Jul 22, 2013 - Jul 27, 2013** | **Live Event** |
| **SEC528 SANS Training Program for the CompTIA® New Advanced Security Practitioner (CASP) Certification** | **Washington, DCUS** | **Jul 22, 2013 - Jul 26, 2013** | **Live Event** |
| **SANS San Francisco 2013** | **San Francisco, CAUS** | **Jul 29, 2013 - Aug 03, 2013** | **Live Event** |
| **SANS SEC 560: Network Penetration Testing @ Bangalore 2013** | **Bangalore, IN** | **Aug 05, 2013 - Aug 10, 2013** | **Live Event** |
| **SANS Boston 2013** | **Boston, MAUS** | **Aug 05, 2013 - Aug 10, 2013** | **Live Event** |
| **Critical Security Controls Summit** | **Washington, DCUS** | **Aug 12, 2013 - Aug 18, 2013** | **Live Event** |
| **Industrial Control Systems Security Training - DC** | **Washington, DCUS** | **Aug 12, 2013 - Aug 16, 2013** | **Live Event** |
| **SANS Thailand 2013** | **Bangkok, TH** | **Aug 19, 2013 - Aug 31, 2013** | **Live Event** |
| **SANS Virginia Beach 2013** | **Virginia Beach, VAUS** | **Aug 19, 2013 - Aug 30, 2013** | **Live Event** |
| **Mobile Device Security Summit 2013** | **OnlineCAUS** | **May 30, 2013 - Jun 06, 2013** | **Live Event** |
| **SANS OnDemand** | **Books & MP3s OnlyUS** | **Anytime** | **Self Paced** |