Elliptic Curve Cryptography

Vivek Kapoor
Department of Computer Engineering
Delhi College of Engineering

Vivek Sonny Abraham
Department of Computer Engineering
Delhi College of Engineering

Ramesh Singh
National Informatics Centre
Government Of India

Abstract

This paper describes the Elliptic Curve Cryptography algorithm and its suitability for smart cards.

1 Information Security

Information security is essential for today's world since, for profitable and legal trading, confidentiality, integrity and non-repudiability of the associated information are necessary. This can be done using cryptographic systems. Intgerated cryptographic systems satisfy all the above-mentioned requirements. Desired properties of a secure communication system may include any or all of the following[wik, PVO96]:

Confidentiality Only an authorized recipient should be able to extract the contents of the encoded data, in part or whole.

Integrity The recipient should be able to establish if the message has been altered during transmission.

Authentication The recipient should be able to identify the sender, and verify that the purported sender actually sent the message.

Non-Repudiation The sender should not be able to deny sending the message, if he actually did send it.

Anti-replay The message should not be allowed to be sent to multimple recipients, without the sender's knowledge.

Proof of Delivery The sender should be able to prove that the recipient received the message.

2 History

Cryptography has been in use for centuries now, and the earliest ciphers were either used transposition or substitution, and messages were encoded and decoded by hand. However, these schemes satisfied only the basic requirement of confidentiality. In more recent times, with the invention of processing machines, more robust algorithms were required, as the simple ciphers were easy to decode using these machines, and moreover they did not have any of the afore mentioned properties. Secure data communication became a necessity in the 20^{th} century and a lot of research was done in this field by government agencies, during and following the world-wars. The most famous machine of this time, Enigma was an electro-mechanical device which was used by the German Army.

2.1 Symmetric Algorithms

The first secret key-based cryptographic algorithms worked on the symmetric algorithms. They assumed that both communicating parties shared some secret information, which was unique to them, much like the older *One Time Pads*. Using this secret information, also called a *key*, the sender encrypted[1] the data, and the recipient was able to decrypt. Suppose Alice wants to send a message m to Bob, and assume

[1] encrypt-encipher-encode and decrypt-decipher-decode are used interchangeably

that they both have already shared a key k. Alice encrypts m using the shared key k to get the cipher text.

$$\mathbb{C}^*{}_{(k,m)} = \mathbf{E}^\dagger{}_k(m) \quad (1)$$

Bob can then decrypt this message using his copy of the key k, and extract the original message m.

$$\mathbf{D}^\ddagger{}_k(\mathbb{C}_{(k,m)}) = \mathbf{D}_k(\mathbf{E}_k(m)) = m \quad (2)$$

This technique though simple and easy to implement, has obvious drawbacks, some of which are listed here:

- A shared secret key must be agreed upon by both parties.
- If a user has n communicating partners, then n secret keys must be maintained, one for each partner.
- Authenticity of origin or receipt cannot be proved because the secret key is shared.
- Management of the symmetric keys becomes problematic.

2.2 Public Key cryptography

The concept of Public Key cryptography(PKC) was first introduced by Di e and Hellman in 1976, in their seminal paper, New Directions in Cryptography [DH76]. This paper also addressed the issue of key exchange, based on the intractability of the discrete logarithm problem. In a public key cryptosystem, each user has a pair of keys, one published publicly, known as the *public key*, and the other known as a *private key*, is stored in a secure location. Public key cryptosystems rely on the existence of a trapdoor function, which makes decoding possible given the knowledge of the private key corresponding to the public key for encryption. Considering a case analogous to the one described in the case of symmetric keys, whereby Alice wishes to send a message m to Bob. the following steps will accomplish the task:

1. Alice passes the message m and Bob's public key $_B$** to an appropriate encryption algorithm to construct the encrypted message.

$$\mathbb{C}_{(\Sigma_B,m)} = \mathbf{E}_{\Sigma_B}(m) \quad (3)$$

2. Alice transmits the encoded message to Bob.
3. Bob decrypts the encrypted message received by him, using his private key Δ_B§ and the appropriate decryption algorithm.

$$\mathbf{D}_{\Delta_B}(\mathbb{C}_{(\Sigma_B,m)}) = \mathbf{D}_{\Delta_B}(\mathbf{E}_{\Sigma_B}(m)) = m \quad (4)$$

Bob is assured that the data he recieved is not tampered with or leaked, as only his private key can decrypt the data. Similarly Bob can send data to Alice using her public key $_A$. The PKC scheme also satisfies the Non-Repudiation and Authenticity by using innovative techniques such as Digital Signatures[Sch95].

3 Smart Cards

3.1 Basics

A smart card, chip card, or integrated circuit(s) card (ICC), is defined as any pocket-sized card with embedded integrated circuits. Although there is a diverse range of applications, there are two broad categories of ICCs. Memory cards contain only non-volatile memory storage components, and perhaps some specific security logic. Microprocessor cards contain memory and microprocessor components. The standard perception of a *smart card* is a microprocessor card of credit-card dimensions (or smaller, e.g. the GSM SIM card) with various tamper-resistant properties (e.g. a secure cryptoprocessor, secure file system, human-readable features) and is capable of providing security services (e.g. confidentiality of information in the memory). Not all chip cards contain a microprocessor (eg. the memory cards), therefore not all chip cards are necessarily also smart cards[wik].

*represents the cipher text corresponding to message m and key k

†represents the Encryption function

‡represents the Decryption function

**Σ_χ represents the published public key of user χ

§Δ_τ represents the secure private key of user τ

3.2 Application

Smartcards were invented and patented in early 1970's, but the first mass use of smartcards was made in 1983, in French pay-phones. A major boom in Smartcard use came in 1990's, with their introduction as *SIM cards* in mobile phones. They are commonly in use now. Smartcard technology is an industry standard defined and controlled by the Joint Technical Committee 1(JCT1) of the International Standards Organization (ISO) and the International Electronic Committee (IEC). The series of international standards ISO/IEC 7816, introduced in 1987 with the latest update in 2003, defines various aspects of a smartcard, including physical characteristics, physical contacts, electronic signals and transmission protocols, commands, security architecture etc. Smartcards don't contain a battery, and become active only when connected with a card reader. When connected, after a reset sequence, the card remains passive, waiting to receive a command request from a client(host) application. Smartcards can be contactless (based on Radio Frequency ID tags), or can have a standard 8-pin contact[Ort03].

Today smartcards are used for various applications all over the world including Banking, Medical records, GSM SIM cards, Identification and cryptographic services. They have storage and processing capability, and are convenient to carry around, and as the processing power and memory capacity of smartcards improves, their range of applications is expanding as well.

3.3 Java Card

Java Card technology adapts the Java platform for use on smart cards and other devices whose environments are highly specialized, and whose memory and processing constraints are typically more severe than those of J2ME devices. On a Java Card platform multiple applications from di erent vendors can co-exist securely. Java Cards are capable of running Java byte codes, and upto 3 applets at once. A major advantage of running downloadable applets is that in case of a security breach, the user only need to download and write a new applet onto his/her Java Card, instead of destroying the Card and waiting for a new one to be shipped to him/her. A typical Java Card device has an 8- or 16-bit CPU running at 5 MHz, with 2K of RAM and more than 32K of non-volatile memory (EEPROM or Flash). High performance smartcards come with a separate processor and cryptographic chip and memory for encryption, and some come with a 32-bit CPU.

4 Current technology - RSA

RSA stands for Rivest, Adleman and Shamir, who devised this algorithm in 1977 at MIT. RSA is the most widely used public-key encryption scheme today. The US patent on the RSA algorithm expired in 2000, but as the algorithm was already published prior to patent application, it precluded patents elsewhere.

The security of the RSA cryptosystem is based on two mathematical problems: the problem of factoring very large numbers, and the RSA problem. Both of these problems are hard, i.e., no e cient algorithm exists for solving them.

The RSA problem is defined as the task of taking e^{th} roots *modulo* a composite n: recovering a value m such that $m^e = c \pmod{n}$, where (e, n) is the public key and c is the ciphertext. Currently the most promising approach to solving the RSA problem is to factor the modulus n. With the ability to recover prime factors, an attacker can compute the secret exponent d from a public key (e, n), then decrypt c using the standard procedure. To accomplish this, an attacker factors n into p and q, and computes $(p-1)(q-1)$ which allows the determination of d from e. No polynomial-time method for factoring large integers on a classical computer has yet been found, but it has not been proven that none exists[wik].

4.1 Basic Algorithm

4.1.1 Key Generation

Suppose Alice and Bob are communicating over an insecure (open) transmission medium, and Alice wants Bob to send her a private (or secure) message [wik].

Using RSA, Alice will take the following steps to generate a public key and a private key:

1. Choose two large numbers prime numbers p and q such that $p \neq q$, randomly and independent of each other.
2. Compute
$$n = pq \tag{5}$$
3. Compute the totient
$$\phi(n) = (p-1)(q-1) \tag{6}$$
4. Choose an integer e such that $1 < e < \phi(n)$ which is coprime to $\phi(n)$.
5. Compute d such that
$$de \equiv 1 \pmod{\phi(n)} \tag{7}$$

The **public key** consists of:

- n, the modulus, and
- e, the public exponent (sometimes *encrpytion exponent*)

The **private key** consists of:

- n, the modulus, and
- e, the private exponent (sometimes *decryption exponent*), which must be kept secret.

Alice transmits the public key to Bob, and keeps the private key secret. p and q are sensitive since they are the factors of n, and allow computation of d given e.

4.1.2 Encrypting Messages

Suppose Bob wishes to send a message M to Alice. He turns M into a number $m < n$, using some previously agreed-upon reversible protocol known as a padding scheme. Bob now has m, and knows n and e, which Alice has announced. He then computes the ciphertext c corresponding to m:

$$c = m^e \pmod{n} \tag{8}$$

Bob transmits c to Alice.

4.1.3 Decrypting Messages

Alice receives c from Bob, and knows her private key d. She can recover m from c by the following procedure:

$$m = c^d \pmod{n} \tag{9}$$

Given m, she can recover the original message M. The decryption procedure works because

$$c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}. \tag{10}$$

Now, since

$$ed = 1 \pmod{p-1} \tag{11}$$

and

$$ed = 1 \pmod{q-1}, \tag{12}$$

Fermat's little theorem yields

$$m^{ed} \equiv m \pmod{p} \tag{13}$$

and

$$m^{ed} \equiv m \pmod{q}. \tag{14}$$

Since p and q are distinct prime numbers, applying the Chinese remainder theorem to these two congruences yields

$$m^{ed} \equiv m \pmod{pq}. \tag{15}$$

Thus,

$$c^d \equiv m \pmod{n}. \tag{16}$$

4.2 Issues with RSA

As of 2005, the largest number factored using general purpose methods is 663-bits long, using state of the art distributed methods. Experts feel that 1024-bit keys may become breakable in the near future(though disputed). 256-bit length keys are breakable in a few hours using a personal computer. The current recommended key-length is 2048-bits[wik]. Though this length may be insignificant for most personal computers in use, it causes low processing power portable devices like smartcards to become ine cient. There are constraints on processor word length, available memory and clock speeds in these devices. As the need for portable and secure identification slowly becomes a necessity, and as RSA key sizes will increase

in proportion to the processor power available, there arises a need to devise a scheme which provies the same level of cryptographic security with smaller key lengths. ECC is one such scheme, described in the following section.

5 ECC

Elliptic Curve Cryptography is an approach to public-key cryptography, based on elliptic curves over finite fields. The technique was first proposed individually by Neal Koblitz and Victor Miller in 1985. The ECC is based on the *Elliptic Curve Discrete Logarithm* problem, which is a known NP-Hard problem. An elliptic curve is defined by the equation,

$$y^2 + xy = x^3 + ax + b^{\dagger} \tag{17}$$

A brief introduction to the mathematics required for elliptic curves, and their usage is given in the following section.

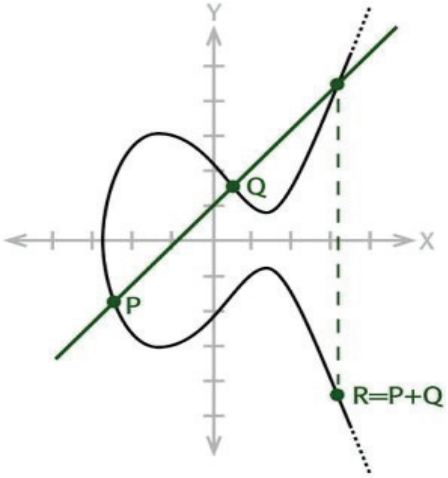

Figure 1: Elliptic curve showing the operation $\mathbf{P} + \mathbf{Q} = \mathbf{R}$. (See Appendix B)

[†]can also be used in the form

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

5.1 Basic Algorithm

5.1.1 Operations on Elliptic Curves

The crucial property of an elliptic curve is that we can define a rule for *adding* two points which are on the curve, to obtain a third point which is also on the curve. This addition rule satisfies the normal properties of addition. The points and the addition law form a finite Abelian group.[Bar97]

For addition to be well defined for any two points, we need to include an extra *zero* point $\mathbf{0}$, which does not satisfy the elliptic curve equation. $\mathbf{0}$ is taken to be a point of the curve. The order of the curve is the number of distinct points on the curve, including the zero point.

Having defined addition of two points, we can also define multiplication $k\mathbf{P}$ where k is a positive integer and $\mathbf{P}$ is a point as the sum of k copies of $\mathbf{P}$.

$$\therefore 2\mathbf{P} = \mathbf{P} + \mathbf{P}$$

5.1.2 Cryptography

Alice, Bob, Cathy, David...agree on a (non-secret) elliptic curve and a (non-secret) fixed curve point $\mathbf{F}$. Alice chooses a secret random integer A_k which is her secret key, and publishes the curve point $\mathbf{A}_P = A_k\mathbf{F}$ as her public key. Bob, Cathy and David do the same.

Now suppose Alice wishes to send a message to Bob. One method is for Alice to simply compute $A_k\mathbf{B}_P$ and use the result as the secret key for a conventional symmetric block cipher (say DES). Bob can compute the same number by calculating $B_k\mathbf{A}_P$, since

$$B_k\mathbf{A}_P = B_k \cdot (A_k\mathbf{F}) = A_k \cdot (B_k\mathbf{F}) = A_k\mathbf{B}_P. \tag{18}$$

The security of the scheme is based on the assumption that it is di cult to compute k given $\mathbf{F}$ and $k\mathbf{F}$.

5.1.3 Choosing the Fixed Curve

A finite field is first chosen (see Appendix A). If the field is $\mathbf{GF}(p)$ where p is a large prime, the xy term

is omitted, leaving us with (see Equation 17)

$$y^2 = x^3 + ax^2 + b, \text{ where } 4a^3 + 27b^2 \neq 0. \quad (19)$$

If the field is $\mathbf{GF}(2^m)$, then we include the xy term to get

$$y^2 + xy = x^3 + ax^2 + b, \text{ where } b \neq 0. \quad (20)$$

Fields $\mathbf{GF}(p^m)$ with both $p > 2$ and $m > 1$ are not considered here.

5.1.4 Choosing the Fixed Point

For any point $\mathbf{P}$ on a elliptic curve in the $\mathbf{GF}(p^m)$,

$$\lim_{k \to \infty} k\mathbf{P} \to \mathbf{0}.$$

For some a and b, $b > a$, we will have $a\mathbf{P} = b\mathbf{P}$. This implies $c\mathbf{P} = \mathbf{0}$ where $c = b - a$. The least c for which this is true is called the *order of the point*, and c must divide the *order of the curve*.

For good security, the curve and fixed point are chosen so that the order of the fixed point $\mathbf{F}$ is a large prime number. This is determined from the order of the curve, which is done from *Schoof's Algorithm*[IKNY98]. For good security, the order of the fixed point should also satisfy the **MOV** condition to prevent certain possible attacks.

As far as is known, with the above provisions, if the order of the fixed point $\mathbf{F}$ is an n-bit prime, then computing k from $k\mathbf{F}$ and F takes roughly $2^{\frac{n}{2}}$ operations.

This is what makes the use of elliptic curves attractive – it means that public keys and signatures can be much smaller than with RSA for the same predicted security.

5.2 Advantages over RSA

5.2.1 Security

The main advantage ECC has over RSA is that the basic operation in ECC is point addition (see Appendix B), which is known to be computationally very expensive. This is one of the reasons why it is very unlikely that a general sub-exponential attack on ECC will be discovered in the near future, though ECC has a few attacks on a few particular classes of curves. These curves can be readily distinguished and can be avoided. On the other hand, RSA already has a known sub-exponential attack which works in general. Thus, to maintain the same degree of security, in view of rising computing power, the number of bits required in the RSA generated key pair will rise much faster than in the ECC generateed key pair, as seen in table 1.

Menezes and Jurisic, in their paper [JM97], said that to achieve reasonable security, a 1024-bit modulus would have to be used in a RSA system, while 160-bit modulus should be su cient for ECC.

Time to break (in MIPS-years)	**RSA key-size** (in bits)	**ECC key-size** (in bits)
10^4	512	106
10^8	768	132
10^{11}	1024	160
10^{20}	2048	210
10^{78}	21000	600

Table 1: Comparison of strength of RSA and ECC

Most attacks on ECC are based on attacks on similar discrete logarithm problems, but these work out to be much slower due to the added complexity of point addition. Also, mathods to avoid each of the attacks have already been designed.[Pie00]

5.2.2 Space Requirements

Due to increasing computation required for higher bit encryption, more transistors are required onboard the smart card to perform the operation. This leads to an increase in area used for processor. Using ECC, the number of transistors can be cut back on since the numbers involved are much smaller than an RSA system with as similar-level security.

Also, the bandwidth requirements for both of the systems is the same when the messages to be signed are long, but ECC is faster when the messages are short. This is more relevant, since PKC is used to transmit mostly short messages, e.g. session ids.

5.2.3 Efficiency

Both methods can be made faster – in RSA, by using smaller public exponent, though this holds a greater security risk and in ECC, some results of the calculation can be stored beforehand. Certicom, a Canadian company, has been studying and promoting the ECC system since the early '80s. Some of their results of fast implementations of ECC compared to RSA are given in table 2.

Function	**ECC 163-bit** (in ms)	**RSA 1024-bit** (in ms)
Key Generation	3.8	4708.3
Sign	2.1(ECNRA) 3.0(ECDSA)	228.4
Verify	9.9(ECNRA) 10.7(ECDSA)	12.7

Table 2: Comparison of RSA and ECC

6 Conclusions

In the discussion above, we have seen that ECC is faster, and occupies less meory space than an equivalent RSA system. This means that it is suitable for constrained environments, especially in smartcards, where fast operations are necessary. Though the industry has been excruciatingly slow in adopting the new technique, RSA Security in an article on their website has implicitly agreed that ECC is the way to the future. The di erence in the key-sizes between ECC and RSA will grow exponentially to maintain the same relative strength as compared to the average computing power available.

The one thing working against ECC is that though elliptic curves has been a well-researched field, albeit an esoteric and extremely vast one[2], its cryptographic applications have been noticed only recently. This is the only advantage that RSA has over ECC. RSA has been well-researched and has been the topic of many seminal theses. In fact, the cryptographic use for elliptic curves was only discovered in the process of finding out new attacks on the RSA system.[Len87]

[2]"It is possible to write endlessly on elliptic curves. (This is not a threat.)"– Serge Lang

A Galois Fields

The familiar examples of fields are $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Q}$ and $\mathbb{Z}$ (mod p)$\forall p =$ prime numbers. The latter is an example of a *finite field*. The requirements of a field are the operations of addition and multiplication, plus the existence of both additive and multiplicative inverses (except that 0 doesn't have a multiplicative inverse). To put it another way, a field has addition, subtraction, multiplication and division – and these operations always produce a result that is in the field, with the exception of division by zero, which is undefined.

Recall that complex numbers can be defined as $a + b \cdot \iota$ with the *reduction rule* $\iota^2 + 1 = 0$. To multiply complex numbers we treat ι as an *unknown*, collect up powers of ι, and apply the reduction rule to simplify the result. It turns out that this construction works for other *reduction rules* involving higher powers of ι. To avoid confusion, in what follows, t is used instead of ι.

The coe cients of the powers of t can be from any field – but if we take the field to be the $\mathbb{Z}$ (mod p), we get a finite field with p^m elements, where m is the degree of the *reduction rule* – that is the exponent of the highest power of t.

For example, if we set $p = 2, m = 4$, and use the *reduction rule* $t^4 + t + 1 = 0$, we get a field with $2^4 = 16$ distinct elements: $0, 1, t, t+1, t^2, t^2+1, t^2+t, t^2+t+1, t^3, t^3+1, t^3+t, t^3+t+1, t^3+t^2, t^3+t^2+1, t^3+t^2+t, t^3+t^2+t+1$.

This construction works for all p and m, as long as p is prime; in fact every finite field can be constructed in this way; moreover two finite fields with the same number of elements are always isomorphic – that is there is a 1-1 map between them which preserves the addition and multiplication rules. This field is called the Galois Field with p^m elements, denoted by $\mathbf{GF}(p^m)$.

B Addition of Points in GF(p)

For elliptic curves, the operation $\mathbf{P} + \mathbf{Q} = \mathbf{R}$ can be carried out by drawing a chord through $\mathbf{P}$ and $\mathbf{Q}$. This chord intersects the ellipitc curve at a third point. This point is $-\mathbf{R}$. $\mathbf{R}$ is found out by reflecting $-\mathbf{R}$ in the x-axis (see Figure 1).

The operation is denoted by:

$$\begin{aligned} \mathbf{P}+\mathbf{Q} &= \mathbf{R} \text{ or,} \\ (x_{\mathbf{P}}, y_{\mathbf{P}}) + (x_{\mathbf{Q}}, y_{\mathbf{Q}}) &= (x_{\mathbf{R}}, y_m athbf R), \text{ where} \\ x_{\mathbf{R}} &= L^2 - x_{\mathbf{P}} - x_{\mathbf{Q}} \text{ and,} \\ y_{\mathbf{R}} &= L(x_{\mathbf{P}} - x_{\mathbf{R}}) - y_{\mathbf{P}} \text{ and,} \\ L &= \frac{(y_{\mathbf{P}} - y_{\mathbf{Q}})}{(x_{\mathbf{Q}} - x_{\mathbf{P}})} \end{aligned}$$

If $x_{\mathbf{P}} = x_{\mathbf{Q}}$ and $y_{\mathbf{P}} = y_{\mathbf{Q}}$ we must use instead,

$$\begin{aligned} x_{\mathbf{R}} &= L^2 - 2x_{\mathbf{P}} \\ y_{\mathbf{R}} &= L(x_{\mathbf{P}} - x_{\mathbf{R}}) - y_{\mathbf{P}} \\ L &= \frac{(3x_{\mathbf{P}}^2 + a)}{2y_{\mathbf{P}}} \end{aligned}$$

If $x_{\mathbf{P}} = x_{\mathbf{Q}}$ and $y_{\mathbf{P}} = -y_{\mathbf{P}}$ then $\mathbf{R} = \mathbf{0}$ and if either point is $\mathbf{0}$, then the result is the other point. If $\mathbf{P} = \mathbf{Q}$, then the chord reduces to a tangent. It can easily be seen that this operation is commutative, associative and distributive.

References

[Bar97] George Barwood. Elliptic curve cryptography faq v1.12. 1997.

[DH76] Whitfield Di e and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[IKNY98] Tetsuya Izu, Jun Kogure, Masayuki Noro, and Kazuhiro Yokoyama. E cient implementation of schoof's algorithm. In *ASIACRYPT '98: Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security*, pages 66–79, London, UK, 1998. Springer-Verlag.

[JM97] Aleksandar Jurisic and Alfred J. Menezes. Elliptic curves and cryptography. *Dr. Dobb's Journal*, 1997.

[Len87] H. W. Lenstra. Factoring integers with elliptic curves. *Annals of Mathematics*, 126:649–673, 1987.

[Ort03] C. Enrique Ortiz. An Introduction to Java Card Technology. 2003.

[Pie00] Henna Pietiläinen. Elliptic curve cryptography on smart cards. 30 October 2000.

[PVO96] Scott A Vanstone P. Van Oorschot, Alfred J Menezes. *Handbook of Applied Cryptography*. CRC Press, 1996.

[Sch95] Bruce Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.

[wik] Wikipedia.