

Elliptic Curve Cryptosystem and its Applications

G.V.S. Raju and Rehan Akbani*

Department of Electrical Engineering.

*Department of Computer Science

The University of Texas at San Antonio.

San Antonio, TX 78249-0669

Email: raju@utsa.edu

Abstract- *The goal of this research is to develop a basis for utilizing efficient encryption schemes in wireless communications and in devices with low computing power and resources. Elliptic Curve Cryptography (ECC) fits well for an efficient and secure encryption scheme. It is more efficient than the ubiquitous RSA based schemes because ECC utilizes smaller key sizes for equivalent security. A comparative study of ECC with RSA is made in terms of key size, computational power, size of data files and encrypted files. Also, another aim is to design an API to implement ECC encryption /decryption algorithm.*

Key Words – Elliptic Curve, Cryptography, Security.

1. Introduction

The rapid progress in wireless mobile communication technology and personal communication systems has prompted new security questions. Since open air is used as the communication channel, the content of the communication may be exposed to an eavesdropper, or system services can be used fraudulently. In order to have reliable proper security over the wireless communication channel, certain security measures need to be provided. The mobile environment aggravates some of the security concerns and threats. Mobile users will use resources at various locations and may be provided by different service providers. Integrity and confidentiality of information stored on the mobile appliance is another important concern. ATM machines and storage of medical records require an efficient encryption algorithm, which uses low processing power.

The vast majority of the products and standards that use public-key cryptography for encryption and digital signatures use RSA[1]. As we have seen, the bit length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA. This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions. Recently, a competing system that has emerged is elliptic curve cryptosystem (ECC)[3,4].

2. Cryptography with Elliptic Curves

The principal attraction of ECC compared to RSA is that it offers equal security for a far smaller key size, thereby reducing processing overhead. The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, we need to find a “hard problem”. All systems rely on the difficulty of a mathematical problem for their security [6]. To explain the concept of difficult mathematical problem, the notion of an algorithm is required. To analyze how long an algorithm takes, computer scientists introduced the idea of polynomial time algorithms and exponential time algorithms. An algorithm runs quickly if it is polynomial time algorithm, and slowly if it is exponential time algorithm. Therefore, easy problems equate with polynomial time algorithms, and difficult problems equate with exponential time algorithms. When looking for a mathematical problem on which to base a public key cryptographic system, cryptographers search for a problem for which the fastest algorithm takes exponential time. The longer it takes to compute the best algorithm for a problem, the more secure a public key cryptosystem based on that problem will be.

Three types of systems [2] are considered secure and efficient: the Integer Factorization Systems (RSA), the Discrete Logarithm systems (DSA)[5], and the Elliptic Curve System (Elliptic Curve Discrete Logarithm System)[3,4]. In RSA, given an integer n which is the product of two large primes p and q such that

$$n = pq. \quad (1)$$

It is easy to calculate n given p and q but it is difficult to determine p and q given n for large values of n . The U.S. government’s Digital Signature Algorithm (DSA) is based on discrete logarithm problem modulo a prime p . Given an integer g between 0 and $p-1$, and y which is the result of exponentiation of g , we have

$$y = g^x \pmod{p} \text{ for some } x. \quad (2)$$

The discrete logarithm problem modulo p is to determine the integer x for a given pair g and y .

The Elliptic Curve Cryptosystem (ECC), whose security rests on the discrete logarithm problem over the points on the elliptic curve. The main attraction of ECC over RSA and DSA is that the best known algorithm for solving the underlying hard mathematical problem in ECC (the elliptic curve discrete logarithm problem (ECDLP) takes full exponential time. RSA and DSA take sub-exponential time. This means that significantly smaller parameters can be used in ECC than in other systems such as RSA and DSA, but with equivalent levels of security. A typical example of the size in bits of the keys used in different public key systems, with a comparable level of security (against known attacks), is that a 160-bit ECC key is equivalent to RSA and DSA with a modulus of 1024 bits. The lack of a sub-exponential attack on ECC offers potential reductions in processing power and memory size. These advantages are specially important in applications on constrained devices.

In practical terms, the performance of ECC depends mainly on the efficiency of finite field computations and fast algorithms for elliptic scalar multiplications. In addition to the numerous known algorithms for these computations, the performance of ECC can be increased by selecting particular underlying finite fields and/or elliptic curves. For ECC, we are concerned with a restricted form of elliptic curve that is defined over a finite field. Of particular interest for cryptography is what is referred to as the elliptic group mod p , where p is a prime number. This is defined as follows. Choose two nonnegative integers, a and b , less than p that satisfy:

$$4a^3 + 27b^2 \pmod{p} \neq 0. \quad (3)$$

Then $E_p(a, b)$ denotes the elliptic group mod p whose elements (x, y) are pairs of nonnegative integers less than p satisfying:

$$y^2 \equiv x^3 + ax + b \pmod{p} \quad (4)$$

together with the point at infinity O .

The elliptic curve discrete logarithm problem can be stated as follows. Fix a prime p and an elliptic curve.

$$Q = xP \quad (5)$$

where xP represents the point P on elliptic curve added to itself x times. Then the elliptic curve discrete logarithm problem is to determine x given P and Q . It is relatively easy to calculate Q given x and P , but it is very hard to determine x given Q and P .

3. ECC Encryption/Decryption

Several approaches to encryption/ decryption using elliptic curves have been analyzed. This paper describes one of

them. The first task in this system is to encode the plaintext message m to be sent as an x - y point P_m . It is the point P_m that will be encrypted as a cipher text and subsequently decrypted. Note that we cannot simply encode the message as the x or y coordinate of a point, because not all such coordinates are in $E_p(a, b)$. There are approaches to encoding. We developed a scheme that will be reported elsewhere. As with the key exchange system, an encryption/decryption system requires a point G and an elliptic group $E_p(a, b)$ as parameters. Each user A selects a private key n_A and generates a public key

$$P_A = n_A \times G. \quad (6)$$

To encrypt and send a message P_m to B , A chooses a random positive integer x and produces the cipher text C_m consisting to the pair of points[7]

$$C_m = \{xG, P_m + xP_B\} \quad (7)$$

Note that A has used B 's public key P_B . To decrypt the cipher text, B multiplies the first point in the pair by B 's secret key and subtracts the result from the second point:

$$P_m + xP_B - n_B(xG) = P_m + x(n_BG) - n_B(xG) = P_m \quad (8)$$

A has masked the message P_m by adding xP_B to it. Nobody but A knows the value of x , so even though P_B is a public key, nobody can remove the mask xP_B . However, A also includes a "clue," which is enough to remove the mask if one knows the private key n_B . For an attacker to recover the message, the attacker would have to compute x given G and xG , which is hard.

4. Development of API

There is the question about what network layers should security be implemented in. Should it be implemented at the IP layer, or at the transport layer or at the application layer? For this reason it is desirable to create an API that implements encryption and later this API can be used at any layer deemed to be most appropriate for a given circumstance. After analyzing the broad picture, we have developed an Application Programmer's Interface (API) that implements ECC, allowing it to be used in a variety of applications. Fig.1 shows how the API fits into the overall network model. As the model suggests, the ECC API is intended to be used in the security layer to automatically encrypt/decrypt all data that flows to or from the application layer. This model allows the application to be oblivious to encryption issues by designating that responsibility to the security layer. The security layer in turn will depend on the API in order to carry out its task. A major advantage of this model is that existing applications do not have to be rewritten to utilize

the ECC API, instead they can be executed as they are and still benefit from the new encryption scheme. ECC API is implemented in Java language instead of C language for the following reasons:

- 1) Java is portable so the API can be used on virtually any device with computing power and with any operating system.
- 2) The new Java Just-In-Time (JIT) compiler is quoted to compile Java programs with optimizations that allow the programs to run about as fast as programs written in C.

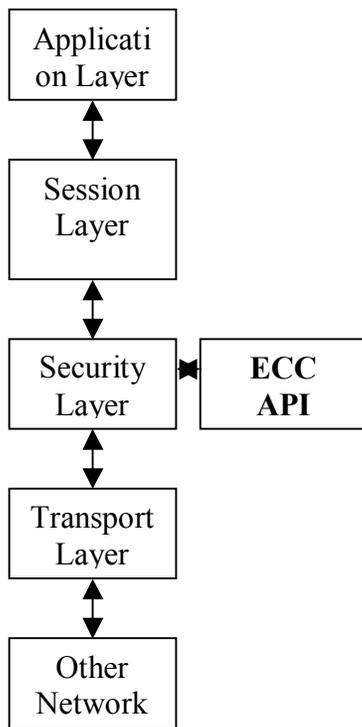


Figure 1- Where the API fits in the overall network

The advantage of portability was the deciding factor in choosing Java. The API will allow calling programs to encrypt or decrypt data using keys provided by the caller. The API will also be capable of generating new keys from scratch that can be subsequently used. The API is designed keeping in mind the low computing resources available to it. We developed a front-end program to demonstrate the functionality of the ECC API. This front-end program utilizes the ECC API to encrypt a plain text data file. The program can be used on computing devices in order to store confidential data securely onto the device.

In addition to encryption and decryption, ECC API can be applied to other applications such as Digital

Signatures, Mutual Authentication, and Secure Data Transmission.

5. Comparison of ECC with RSA

We compare the performance of ECC with RSA in terms of key sizes for the same level of security, data sizes, encrypted message sizes, and computational power.

RSA takes sub-exponential time and ECC takes full-exponential time. For example, RSA with key size of 1024 bits takes 3×10^{11} MIP years with best known attack where as ECC with 160 bit key size takes 9.6×10^{11} MIP years. Therefore, ECC offers same level of security with smaller key sizes. DATA size for RSA is smaller than ECC. Encrypted message is a function of key size and data size for both RSA and ECC. Since ECC key size is relatively smaller than RSA key size, encrypted message in ECC is smaller. As a result, computational power is smaller for ECC.

6. Conclusions

The security of Elliptic Curve Cryptosystem depends on how difficult it is to determine x given xP and P . This is referred to as the elliptic curve logarithm problem. The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method. It has been seen that a considerably smaller key size can be used for ECC compared to RSA. Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA. The results show that ECC is efficient in terms of the size of Data files and Encrypted files.

The above information is useful for wireless communication due to low data rate transmission and for constrained devices because of low power requirements.

We have developed an Application Programmer's Interface (API) that implements ECC, allowing it to be used in a variety of applications. As the model suggests, the ECC API is used in the security layer to automatically encrypt/decrypt all data that flows to or from the application layer. This model allows the application to be oblivious to encryption issues by designating that responsibility to the security layer. The security layer in turn will depend on the API to carry out its task. A major advantage of this model is that existing applications do not have to be rewritten to utilize the ECC API, instead they can be executed as they are and still benefit from the new encryption scheme.

REFERENCES

[1] R.L. Rivest, A. Shamir, and L.M. Adleman, "A

Method for Obtaining Digital Signatures and Public-key Cryptosystems “, Communications of the ACM, Volume 21, pages 120-126, February 1978.

- [2] Certicom Corp., “ An Introduction to Information Security”, Number 1, March 1997.
- [3] N. Koblitz, “ Elliptic Curve Cryptosystems “, Mathematics of Computation., Number 48,pages 203-209,1987.
- [4] V.S. Miller,” Use of Elliptic Curves in Cryptography “, Advances in Cryptology- Proceedings of CRYPTO '85, Springer Verlag Lecture Notes in Computer Science 218, pages 417-426, 1986.
- [5] National Institute of Standards and Technology, “ Digital Signature Standard ‘, FIPS Publication 186, 1993.
- [6] J. Lopez and R. Dahab, “ An Overview of Elliptic Curve Cryptography”, Relatorio Tecnico IC-00-10, May 2000.
- [7] W. Stallings,“ Cryptography and Network Security”, Prentice Hall, Second Edition,1998.