UDC 681.8

Elliptic Curve Cryptosystem

●Naoya Torii ●Kazuhiro Yokoyama

(Manuscript received June 6, 2000)

This paper describes elliptic curve cryptosystems (ECCs), which are expected to become the next-generation public key cryptosystems, and also describes Fujitsu Laboratories' study of ECCs. ECCs require a shorter key length than RSA cryptosystems, which are the de facto standards of public key cryptosystems, but provide equivalent security levels. Because of the shorter key length, ECCs are fast and can be implemented with less hardware.
First, we outline ECC and describe a typical digital signature algorithm. Then, we describe our technology for parameter generation of a secure ECC and the implementation of a fast ECC by software and by a digital signal processor. ECCs are expected to enter widespread use as a base technology of electronic information services.

1. Introduction

Various services on open networks, for example, electronic commerce; electronic shops for music, video, and software; and CALS/EDI, are expected to make life more convenient and efficient. Encryption is a base technology used to realize these services. There are two types of encryption technology: secret key cryptosystems and public key cryptosystems. In a secret key cryptosystem, a key for encryption and decryption is shared between the sender and the receiver. In a public key cryptosystem, the sender and the receiver use different keys. Some commonly used public key cryptosystems are the RSA cryptosystem (RSA)[1] and the ElGamal cryptosystem;[2] these were invented in 1978 and 1984, respectively.

The elliptic curve cryptosystem (ECC) was invented by N. Koblitz[3] and by V. Miller[4] independently in 1985 and is expected to become the next-generation public key cryptosystem. A lot of work is being done to learn about its security, and standardization organizations such as the ISO, IEEE, ANSI, and IETF are actively standardizing ECC so it can be put to practical use.

2. Elliptic Curve Cryptosystems (ECCs)

2.1 Public key cryptosystems

A public key cryptosystem employs a pair of different but associated keys. One of these keys is released to the public while the other, the private key, is known only to its owner. It is designed to be computationally intractable to calculate a private key from its associated public key; that is, it is believed that any attempt to compute it will fail even when up-to-date technology and equipment are used.

With a public key cryptosystem, the sender can encrypt a message using the receiver's public key without needing to know the private key of the receiver. Therefore, they are suitable for communication among the general public.

Public key cryptosystems can also be used to make a digital signature. In the RSA, signature generation is done by encrypting the message using the sender's private key. The signature verification is done by decrypting the signed mes-

sage using the sender's public key. If the received message and decrypted message are the same, the signature is correct and the integrity of the signed message is assured. The merit of the signature in a public key cryptosystem is that the verification is processed using only the sender's public key; that is, the sender's private key is not needed for verification.

Public key cryptosystems provide an encryption and digital signature scheme without the need to reveal a private key. Because of this feature, these cryptosystems are considered to be indispensable for secure communication and authentication over open networks.

2.2 Elliptic Curve Cryptosystems

Elliptic curve cryptosystems (ECCs) include key distribution, encryption, and digital signature algorithms. The key distribution algorithm is used to share a secret key, the encryption algorithm enables confidential communication, and the digital signature algorithm is used to authenticate the signer and validate the integrity of the message.

ECCs are based on the addition of rational points on a chosen elliptic curve. An elliptic curve E over a Galois field GF(p), where $p>3$ and is prime, is the set of all (x, y) $(x, y \in \mathrm{GF}(p))$ that satisfy the following equation:

$$\mathrm{E}: y^2 = x^3 + ax + b$$

where $a, b \in \mathrm{GF}(p)$, and $4a^3 + 27b^2 \neq 0$.

The rational points on the elliptic curve E are the points over GF(p) that satisfy the defining equation. If the set of parameters $\{a, b, p\}$ are specified, the number of rational points on the elliptic curve is determined uniquely; this number is called the order of the elliptic curve E and is denoted by #E. It is known that rational points form an additive group in the addition over the elliptic curve shown in **Figure 1**.

Instead of giving a rigorous definition of the addition of two rational points on an ECC over GF(p), we will give an intuitive definition of addition by using an illustrative model of an ECC over real numbers.

When points A and B on the elliptic curve E shown in Figure 1 are added, the result is defined as the point D obtained by inverting the sign of the y-coordinate of point C, where point C is the intersection of E and the line passing through A and B. If A and B are at the same position, the line is the tangent of E at A.

Moreover, an ideally defined point O, namely the point at infinity, is also recognized as a point on E. The sum of the point at infinity and a point P is defined as point P itself.

We define the k scalar multiplication of a point G as the operation by which point G is added to itself k times. We denote the resulting point as kG. We can easily calculate W = kG from a given k and G, but it is computationally difficult to calculate the scalar k from points W and G. If a prime p as large as 160 bits long is selected, we cannot find k within a reasonable time, even if we use the most efficient algorithms known so far and the world's most powerful computers. The problem of calculating k from given points G and W is called "the discrete logarithm problem over the elliptic curve." The security of ECC derives from the difficulty of solving the problem.

Moreover, when a point G on an elliptic curve E is given, there is a minimum positive integer n such that nG = O. Integer n is called the order of

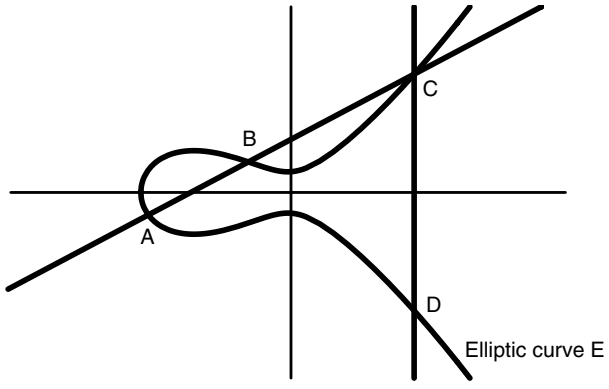

Figure 1
Addition rule over an elliptic curve.

the point G. It is known that n is a divisor of the order of the curve E.

Elliptic curves over a characteristic 2 finite field GF(2^m) which has 2^m elements have also been constructed and are being standardized for use in ECCs as alternatives to elliptic curves over a prime finite field.

2.3 Signature algorithm

We now describe ECDSA[5] as an example of a digital signature algorithm.

In a digital signature algorithm, the sender sends a message with the sender's own unique signature and the receiver validates the received signature. Instead of providing a signature for the entire message, the message is first shortened to a fixed length by a hash function, then a short signature that is valid over the whole message is generated.

In ECC, the system parameters such as the prime p, elliptic curve E, base point G = (x, y), and order r of the point G need to be shared between the sender and the receiver. Let s, where $1 \leq s \leq r-1$, be the private key of the sender and W = sG the public key of the sender.

The signature is generated as follows:

Hash the message, and obtain the hash value f = hash(m).
Generate a random number u, where $1 \leq u \leq r-1$.
Calculate V = uG = (x_v, y_v) and $c = x_v \bmod r$.
Calculate $d = u^{-1}(f + sc) \bmod r$.
Output (c, d) as a signature of m.

The sender then sends the message m and the signature (c, d).

The signature is validated as follows:

The receiver receives the message m and the signature (c, d). Then, the receiver performs the following procedure to validate the signature:
Hash the message, and obtain the hash value f = hash(m).
Calculate $h = d^{-1} \bmod r$.
Calculate $h_1 = fh \bmod r$ and $h_2 = c$h $\bmod r$.
Calculate P = h_1G + h_2W = (x_P, y_P) and $c' = x_P \bmod r$.
If $c = c'$, then the signature is valid. Otherwise, it is invalid.

2.4 Merit of ECCs

The merit of ECCs is that compared with RSA cryptosystems they can provide the same security level with a shorter key length. Because of this mathematical property, ECCs are faster and require less hardware than RSA. The security of an ECC, however, depends not only on the length of the key, but also on the elliptic curve parameters. In general, it takes a long time to generate secure parameters. So, faster parameter generation is important for practical implementation of an ECC. We describe our research concerning the security of parameters in the next chapter.

3. Security

3.1 Concept of security evaluation

In general, the security of a cryptosystem is evaluated by the amount of time needed to break it. "Breaking a cryptosystem" means finding the private key used to encrypt a message. The method used to break a cryptosystem is called the "attacking method." Normally, the time needed to break a practical cryptosystem is never actually obtained, because a cryptosystem that can be broken in a reasonable amount of time would not be considered for practical use.

Instead, the amount of time needed to break a cryptosystem is a theoretical estimate of the average time needed to break a cryptosystem by a given attacking method. If there are multiple attacking methods, the time required by the most efficient method would be taken as the time needed to break the cryptosystem.

A security evaluation based on the theoretical estimate of average time is valid for the general case, but it is clear that such an evaluation is invalid for special cases.

3.2 General attack for ECCs

As described in Section 2.2 the security of an

ECC depends on the difficulty of solving the discrete logarithm problem over elliptic curves. Two general methods of solving this problem are known. One is the square root method, which is a general method for the discrete logarithm problem. The other is the Silver-Pohlig-Hellman (SPH) method,[6),7)] which factors the order of a curve into small primes and solves the discrete logarithm problem as a combination of discrete logarithms for small numbers.

The square root method is the most general attacking method for the discrete logarithm problem, and its computation time is proportional to the exponent of half the key length; that is, the computation time varies exponentially with respect to the key length. A public key cryptosystem is regarded as being very secure against an attack if the attack takes an exponential amount of time with respect to the key length. From this criterion, we can say that ECCs are very secure against the square root method.

The SPH method is effective only when the order of the curve is expressed as a product of small primes. Otherwise, the computation time is equivalent to that of the square root method. Therefore, for an ECC, if we select the order of the elliptic curve to be a prime or nearly a prime whose factors include a large prime, the computation time needed to break the ECC will vary exponentially. Therefore a high level of security can be achieved.

Now we will compare the security of ECCs with that of RSA. The security of RSA lies in the difficulty of factoring large numbers. The number field sieve is the most effective known method for factoring large numbers, and it takes a sub-exponential amount of computation time with respect to the key length to do that task. Therefore, the best-known attack against RSA takes a sub-exponential amount of time with respect to the key length. An attacking method with a sub-exponential time/key-length relationship takes less time than one with an exponential relationship (and more time than a method with a polynomial relationship). This is why the securities of 1024-bit RSA and 160-bit ECC are equivalent.

3.3 Evaluation by special attack

The security evaluation of a general attack against an ECC given above is based on the average computation time. However, we cannot evaluate special cases with this evaluation. In fact, special attacks were found by using the special characteristics of special elliptic curves.[8),9)] The special characteristics are determined by the order of the elliptic curve. These special attacks are much stronger than the square-root method.

3.4 Generating a secure curve parameter

Because of the threat of special attacks, it is essential to obtain the parameters of elliptic curves that meet the following basic requirements:

- The order of the curve must be prime or nearly prime.
- The curve must be immune to special attacks.

Both of these requirements concern the order of the curve, #E. That is, the security of an elliptic curve depends primarily on its order. Therefore, to make an ECC secure, we must first find curves which have an order satisfying the above requirements. At present, two methods[10),11)] are proposed to find good curves:

1) Generate a curve randomly, count its order, and select a curve satisfying the criteria.

2) Select an order satisfying the requirements, and then generate a curve of the selected order.

For method 1), the Schoof algorithm[10)] is used. Theoretically, the computation time for the Schoof algorithm is polynomial with respect to the key length. However, in practice it takes a long time to calculate if we implement it directly in the present computational environment.

For method 2), the complex multiplication algorithm (CM algorithm)[11)] is used. The order of computation for the general CM algorithm is exponential with respect to the key length, so it is

hard to calculate. Therefore, in practice, the order of the curve is selected to have special characteristics so that it can be calculated efficiently. However, as described in the previous section, there is the possibility of attacks using the special characteristics. We think it doubtful that curves generated by method (2) above will be safe in the future. We therefore adopted method (1) and improved Schoof's algorithm.

3.5 Our technology for generating good ECC parameters

We devised an efficient method, called the ICS method, to calculate the order of an elliptic curve.[12),13)] Then, we developed software based on this method that can deal with two major types of finite fields (i.e., prime fields and characteristic 2 finite fields) and can produce a set of elliptic curve parameters for an ECC. To use an ECC in practice, the bit length of the base field should be from 160 to 240 bits. Our software calculates these parameters within a reasonable time. **Table 1** shows the measured times for parameter generation on a Pentium II 400 MHz PC running under Windows NT 4.0 and Risa/Asir.[14)]

With this technology, we can generate as many secure curves as we need, so it is easy to construct an ECC system.

4. Implementation

To be of practical use, an ECC must have a high operating speed. In this chapter we briefly describe how we achieved a fast ECC system through software and hardware.

4.1 Software engine

Various methods for achieving an efficient ECC have been proposed, and basic ECC algorithms are described in various standards. We developed a general-purpose software engine for the server which can deal with an ECC with any elliptic curve parameters. The engine has been made fast using various speed-up techniques and conforms with IEEE P1363,[5)] characteristic 2 finite fields (normal base and polynomial base), and prime finite fields.

Moreover, our library can operate on elliptic curve cryptosystems having arbitrary bit lengths up to the memory limit of the platform and can deal with any elliptic curve parameters. The performance of our software engine on a Pentium Pro 200 MHz PC running under Windows NT 4.0 is shown in **Table 2**.

4.2 Hardware engine

For server systems, we have developed an efficient hardware ECC engine based on a digital signal processor (DSP) and devised new, fast implementation methods suitable for the DSP.[15)] We improved modular multiplication (i.e., multiplication with large moduli) and elliptic doubling (i.e., doubling of a point on an elliptic curve) to speed up the implementation. For modular multiplication, we devised a new implementation method for Montgomery multiplication[16)] that is suitable for pipeline processing. For elliptic doubling, we devised an improved method for computing the number of multiplications and additions. We have implemented ECDSA using the latest DSP; it works on an elliptic curve cryptosystem having

Table 1
Parameter generation time.

	Characteristic 2 finite fields (Polynomial base)		Prime finite fields ($p = 2^n$-α)	
Bit length	160	239	160	224
Generation time (s)	266	3783	367	2566

Table 2
Performance of software library.

Base field	Prime finite fields		Characteristic 2 finite fields			
			polynomial base		normal base	
Bit length	160	239	163	239	162	191
Signature generation (ms)	4.1	10	8.4	17	7.7	10
Signature verification (ms)	18	43	34	66	30	37

an arbitrary bit length of up to 320 bits over a prime finite field. In addition, this hardware engine can work on elliptic curves with arbitrary curve parameters in the same way that the software library does. The main specifications of the hardware engine are shown in **Table 3**.

5. Discussion

Special attacks for special elliptic curves are constantly being discovered, and the security of special curves has been actively discussed in recent years. Especially, we think that whether the curves generated by the CM method are secure should be examined intensively. In addition, we should keep in mind that the evaluation of security changes when a stronger attack is found, because the currently evaluated security level is based on the currently known attacks. Therefore, it is desirable to construct a theoretical evaluation that also considers unknown attacks.

Before ECC comes into widespread use in electronic commerce protocols and various services, we can expect that discussions will be held to decide on its specifications, format, and other details.

6. Conclusion

We have briefly described ECC, which is a promising candidate for the next-generation public key cryptosystem. Although ECC's security has not been completely evaluated, it is expected to come into widespread use in various fields in the future because of its compactness and high performance when it is hardware-implemented.

Table 3
Specifications of hardware engine.

DSP			TMS320C601
Clock			200 MHz
Firmware size			32 Kbytes
Performance	Signature generation	160-bit	1.1 ms
		239-bit	2.7 ms
	Signature verification	160-bit	3.8 ms
		239-bit	10 ms

References

1) R. L. Rivest, A. Shamir, and L. M. Adleman: A method for obtaining digital signatures and public-key cryptosystems. *Commu. ACM*, **21**, pp.120-126 (1978).
2) T. ElGamal: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Info. Theory*, **31**, pp.469-472 (1985).
3) N. Koblitz: Elliptic curve cryptosystems. *Math. Comp.*, **48**, pp.203-209 (1987).
4) V. S. Miller: Use of elliptic curves in cryptography. Advances in Cryptology - Crypto'85, Lecture Notes in Computer Science 218, pp.417-426, Springer-Verlag, 1986.
5) IEEE P1363/D13 (Draft Version 13) Standard Specifications for Public Key Cryptography. 1999. *http://grouper.ieee.org/groups/1363/index.html*
6) S. C. Pohlig and M. E. Hellman: An improved algorithm for computing logarithms over GF(p) and its cryptographic significance, *IEEE Trans. Info. Theory*, **24**, pp.106-110 (1978).
7) N. Koblitz: A Course in Number Theory and Cryptography. Graduate Texts in Mathematics, (2nd edition), 114, pp.102-103, Springer-Verlag, 1994.
8) A. J. Menezaes, T. Okamoto, and S. A. Vanstone: Reducing elliptic curve logarithms to a finite field. *IEEE Trans. Info. Theory*, **39**, pp.1639-1646 (1993).
9) T. Satoh and K. Araki: Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curve. *Comm. Math.* Univ. Scncti, Pauli, **47**, pp.81-92 (1998).
10) R. Schoof: Elliptic curves over finite fields and the computation of square roots mod p. *Math. Comp.*, **44**, pp.483-494 (1998).
11) O. Atkin and F. Morain: Elliptic curves and primality proving. *Math. Comp.*, **61**, pp.29-68 (1993).
12) T. Izu, J. Kogure, M. Noro, and K. Yokoyama:

Efficient implementation of Schoof's algorithm. Advances in Cryptology-ASIACRYPT'98, Lecture Notes in Computer Science, 1514, pp.66-79, Springer-Verlag, 1998.

13) T. Izu, J. Kogure, and K. Yokoyama: Efficient Implementation of Schoof's Algorithm in case of characteristic 2. PKC 2000, Lecture Notes in Computer Science, 1751, Springer-Verlag, pp.210-222, 2000.

14) M. Noro and T. Takeshima: Risa/Asir – a computer algebra system. Proceedings of ISSAC'92, ACM Press, 387-396, 1992.

15) K. Ito, M.Takenaka, N. Torii, S. Temma, and Y. Kurihara: Fast implementation of public-key cryptography on a DSP TMS320C6201. First International Workshop, CHES'99, Lecture Notes in Computer Science 1717, pp.61-72, Springer-Verlag, 1999.

16) P. L. Montgomery: Modular Multiplication without Trial Division. *Math. Comp.*, **44**, pp.519-521 (1985).

Naoya Torii received the B.E. and M.E. degrees in Communication Engineering from Osaka University, Suita, Japan in 1981 and 1983, respectively. He joined Fujitsu Laboratories Ltd., Kawasaki, Japan in 1983 and has been engaged in research and development of voice scramblers, high-speed encryption engines, and information security systems. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan, and the IEEE. He is currently working at Fujitsu Laboratories Ltd., Akashi, Japan.

Kazuhiro Yokoyama received the B.S. and M.S. degrees in Mathematics from the University of Tokyo, Tokyo, Japan in 1981 and 1983, respectively. He then received the Ph.D. from Kyushu University, Fukuoka, Japan in 1991. He joined the Fujitsu International Institute for Advanced Study of Social Information Science, Numazu, Japan in 1985 and has been engaged in research of computer algebra and its applications. He is a member of the Mathematical Society of Japan and the Japan Society for Symbolic and Algebraic Computation. He is currently working at Fujitsu Laboratories Ltd., Kawasaki, Japan.