The Elliptic Curve Cryptosystem

CURRENT PUBLIC-KEY CRYPTOGRAPHIC SYSTEMS

Published: April 1997

Updated: July 2000

A Certicom Whitepaper

The Elliptic Curve Cryptosystem (ECC) provides the highest strength-per-bit of any cryptosystem known today. This paper provides an overview of current public-key cryptographic systems and compares the ECC with other well known systems.

Contents

Please send comments and inquiries to:

Certicom
5520 Explorer Drive 4th Floor
Mississauga, Ontario, Canada L4W 5L1
Phone: (905) 507-4220

FAX: (905) 507-4230

info@certicom.com
http://www.certicom.com

1. Introduction

Since the invention of public-key cryptography in 1976 by Whitfield Diffie and Martin Hellman [10], numerous public-key cryptographic systems have been proposed. All of these systems rely on the difficulty of a mathematical problem for their security.

Before cryptographic systems and the corresponding mathematical problems can be discussed, the difficulty of a problem must be defined. What does it mean for a mathematical problem to be difficult? To explain this concept, the notion of an **algorithm** is required. An algorithm is a process which describes the steps to take to solve a problem. For example, in high school everyone is taught an algorithm for adding two numbers - simply a sequence of steps which takes as input two numbers a and b, to be added, and outputs their sum $a+b$. Now a mathematical problem is **difficult** if the fastest algorithm to solve the problem takes a long time relative to the input size.

To analyze how long an algorithm takes, computer scientists introduced the idea of **polynomial time** algorithms and **exponential time** algorithms. Roughly speaking, an algorithm runs quickly relative to the size of its input if it is a polynomial time algorithm, and slowly if it is an exponential time algorithm. Therefore, easy problems equate with polynomial time algorithms, and difficult problems equate with exponential time algorithms.

It is important to notice the words “relative to the input size” in the definition of polynomial time and exponential time algorithms. All problems are straightforward to solve if the input size is very small, but we are interested in how much harder a problem gets as the size of the input grows. For example, adding 2 and 12 to get 14 is straightforward, as is factoring 15 as 3×5. However, addition is an example of an easy problem, because there is an algorithm to add numbers which runs in polynomial time, meaning that it would not take very long to add two enormous numbers. On the other hand, factoring is a hard problem because, in general, factoring a large number takes a very long time.

Thus, when looking for a mathematical problem on which to base a public-key cryptographic system, cryptographers are searching for a problem for which the fastest algorithm takes exponential time. In broad terms, 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.

Over the years, many of the proposed public-key cryptographic systems have been broken (that is - proven to be based on an easier problem than first thought), and many others have proved impractical. Today, only three types of systems should be considered both secure and efficient. The systems, classified according to the mathematical problem on which they are based, are: the

Integer Factorization systems (of which RSA is the best known example), the **Discrete Logarithm systems** (such as the U.S. government's DSA), and the **Elliptic Curve Cryptosystem** (also defined as the Elliptic curve Discrete Logarithm System).

While the mathematical operation of each of these systems is beyond the scope of this whitepaper, in the following sections the underlying problems themselves will be described in order to give readers a flavor of each type of system. Two problems will be outlined for each type of system: the difficult mathematical problem providing security, and the (relatively easy) problem involved in implementing the systems. Finally, the security and efficiency issues for these cryptosystems will be compared so that some of the rationale behind choosing a particular system can be understood.

2. Integer Factorization Systems

While Diffie and Hellman discovered the concept of public-key cryptography in 1976 [10], they did not propose a practical public-key cryptographic system. The first practical public-key cryptographic system was developed two years later at MIT [19]. This system is called **RSA**, after its inventors: Ron Rivest, Adi Shamir, and Len Adleman.

2.1 Security

RSA is the best known of a family of systems whose security relies on the difficulty of the **integer factorization problem**. Another example is the Rabin-Williams system [17, 20].

The integer factorization problem is defined as follows. Recall that an integer (a whole number) p is **prime** if it is divisible only by 1 and p itself. Now, given an integer n which is the product of two large primes, determine these factors, i.e. find primes p and q such that:

$$p \times q = n.$$

As a small example of the integer factorization problem, consider the integer 15. Here the solution consists of the primes 3 and 5, since:

$$3 \times 5 = 15.$$

An RSA public key consists of a pair (n,e), where e is a number between 1 and n-1, and n is the product of two large primes. It is widely believed that to break RSA in general, the integer factorization problem must be solved for the integer n. Since there is no efficient algorithm for the integer factorization problem, n

can be chosen to be large enough to ensure that the system is secure. Due to that the 512-bit number RSA155, an RSA modulus of 155 decimal digits, was factored in August 1999, to provide even short-term security, given today's computing power, *n* should be at least 230 decimal digits long (230 decimal digits is approximately 760 bits).

2.2 Implementation

RSA, and the other members of the integer factorization family, can be used both for encryption and for digital signatures. To describe the operations used to perform these processes, **modular arithmetic** must first be defined.

Modular addition and **modular multiplication** modulo *n* work just like ordinary addition and multiplication, except that the answer to the calculation is reduced to its remainder on division by *n*, so that the answer always lies between 0 and *n*-1. The phrase "mod *n*" is written after each calculation to denote modular arithmetic. For example:

$$3 \times 5 = 1 \pmod 7$$

because 15 has remainder 1 when divided by 7. **Modular exponentiation** can also be performed. For example:

$$2^4 = 2 \pmod 7$$

because 16 has remainder 2 when divided by 7.

Modular arithmetic plays a central role in the implementation of all three types of public-key cryptosystems. When RSA is used either as an encryption scheme or as a digital signature scheme, exponentiation modulo *n* must be performed. Suppose *m*, a number between 0 and *n*-1, represents a message. Then the modular exponentiation

$$m^x \pmod n$$

must be calculated for some number *x* when *m* is transformed. This modular exponentiation dominates the time taken to perform the transformations involved in RSA, so that the time required to calculate modular exponentiation modulo *n* essentially determines the time required to perform RSA.

In summary, the security of RSA, and the other members of the integer factorization family, rests on the difficulty of the integer factorization problem, and its efficiency rests on the speed of performing exponentiation modulo *n*.

3. Discrete Logarithm Systems

3.1 Security

Another mathematical problem defined in terms of modular arithmetic is the **discrete logarithm problem** modulo a prime p. Fix a prime number p. Then given an integer g between 0 and p-1, and y which is the result of exponentiating g, we have the following relation between g and y:

$$y = g^x \pmod{p}$$

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

Like the integer factorization problem, no efficient algorithm is known in general to solve the discrete logarithm problem modulo p. Taher ElGamal was the first to propose a public-key cryptosystem based on this problem. In fact, ElGamal proposed two distinct systems: one to provide encryption, and one to perform digital signatures [11]. In 1991, Claus Schnorr discovered a variant of the ElGamal digital signature system which offers added efficiency compared to the original system [20]. In turn, the U.S. government's **Digital Signature Algorithm** (DSA)[17] is based on ElGamal's work. These systems are the best-known of a large number of systems whose security is based on the discrete logarithm problem modulo p. The prime p used in discrete logarithm systems should also be at least 230 decimal digits (760 bits) in length to provide short-term security.

3.2 Implementation

As was the case with RSA, modular exponentiation must be performed to operate discrete logarithm systems. In every case, the dominant calculation in each of the transformations is:

$$g^x \pmod{p}$$

for some integer x, and a fixed number g between 0 and p-1.

Therefore, discrete logarithm systems can be categorized as follows: their security relies on the discrete logarithm problem modulo p, and their efficiency on the speed of performing modular exponentiation modulo p.

4. The Elliptic Curve Cryptosystem

In section 3, the discrete logarithm problem modulo p was described in terms of modular arithmetic on the remainders on division by p. This is not the only mathematical structure over which the discrete logarithm problem can be defined. In 1985, Neil Koblitz [14] and Victor Miller [16] independently proposed the **Elliptic Curve Cryptosystem** (ECC), whose security rests on the discrete logarithm problem over the points on an elliptic curve. ECC can be used to provide both a digital signature scheme and an encryption scheme.

4.1 Security

An **elliptic curve**, defined modulo a prime p, is the set of solutions (x,y) to an equation of the form:

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

for two numbers a and b. If (x,y) satisfies the above equation then $P=(x,y)$ is a **point** on the elliptic curve.

In fact, an elliptic curve can also be defined over the finite field consisting of 2^m elements. Such a representation offers extra efficiency in the operation of the ECC.

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

$$P + Q$$

will always be another point on the curve. The elliptic curve discrete logarithm problem can be stated as follows. Fix a prime p and an elliptic curve. xP represents the point P added to itself x times. Suppose Q is a multiple of P, so that

$$Q = xP$$

for some x. Then the **elliptic curve discrete logarithm problem** is to determine x given P and Q.

The security of the ECC rests on the difficulty of the elliptic curve discrete logarithm problem. As was the case with the integer factorization problem and the discrete logarithm problem modulo p, no efficient algorithm is known to solve the elliptic curve discrete logarithm problem. In fact, as is shown in the next section, one of the advantages of the ECC is that the elliptic curve discrete logarithm problem is believed to be harder than both the integer factorization problem and the discrete logarithm problem modulo p. This extra difficulty

makes the ECC the strongest public-key cryptographic system known today. Moderate security can be achieved with the ECC using an elliptic curve defined modulo a prime p that is several times shorter than 230 decimal digits. These issues of security will be discussed further in Section 5.

4.2 Implementation

Just as modular exponentiation determined the efficiency of integer factorization and discrete logarithm systems, so it is the calculation of

$$Q = xP$$

for a point P on the elliptic curve and some integer x which dominates the calculations involved in the operation of the ECC. The process of adding elliptic curve points requires a few modular calculations, so in all three cases the operation of a public-key cryptographic system is dependent upon efficient modular arithmetic. As shown in Section 4.1, the prime p used in the ECC can be smaller than the numbers required in the other types of systems, so another advantage of the ECC is that the modular calculations required in its operation are carried out over a smaller modulus. This leads to a significant improvement in efficiency in the operation of the ECC over both integer factorization and discrete logarithm systems, as we will see in the next section.

Thus, in the case of the ECC, security rests on the elliptic curve discrete logarithm problem, while efficiency is dependent on the fast calculation of xP for some number x and a point P on the curve.

5. Comparison of Public-Key Cryptographic Systems

What are the main issues when comparing public-key cryptographic systems? Without doubt the two major benchmarks are **security** and **efficiency**. Less central concerns such as interoperability and public acceptance will be addressed in Section 6.

5.1 Security

In this article, the emphasis in terms of security is placed on **theoretical security** - breaking a public-key system in general. As a brief caveat, anyone implementing such a system should bear in mind that the greatest practical threats are often more physical. Bribing employees to disclose secret keys,

inadvertently disclosing secret information, and tampering with (supposedly) tamper-proof boxes are among a plethora of practical attacks that occur at the implementation level. These types of attack are implementation-specific and are not addressed further here.

When examining the theoretical security of a public-key cryptographic system, the first question to be asked is: does breaking the system really require solving the underlying mathematical problem? It is reasonable to assume the answer to this question is yes since each of the three types of systems discussed in this whitepaper have undergone public scrutiny during a number of years. In fact, for a number of the systems proposed, a formal mathematical proof has been supplied to show that breaking the cryptosystem really does mean solving the mathematical problem.

Therefore this question can be reduced to simply: which is the hardest problem? The integer factorization problem, the discrete logarithm problem modulo p, or the elliptic curve discrete logarithm problem? Unfortunately there are no mathematical problems for which it can be proven that the best algorithm takes fully exponential time. Hence this discussion must focus on the best algorithms known today to solve these problems.

With each of the three problems, there are good, special-purpose algorithms that solve the problem quickly in certain special instances. For integer factorization, there is a fast algorithm when $n=p\times q$ provided p-1 or q-1 only has small prime factors. Similarly, for the discrete logarithm problem modulo p, there is a fast algorithm provided p-1 only has small prime factors. Finally, the elliptic curve discrete logarithm problem is relatively easy for a small class of elliptic curves, known as **supersingular elliptic curves** and also for certain **anomalous elliptic curves**. Fortunately, in each case, the "weak" instances of the problem are easily identified, so an implementation merely checks that the specific instance selected is not one of the class of easy problems. This approach avoids attacks employing these special purpose algorithms.

It remains to consider general-purpose algorithms – those that always succeed in solving the problem. Of the three problems, the integer factorization problem and the discrete logarithm problem modulo p both admit general algorithms that run in **sub-exponential time**. These sub-exponential time algorithms mean that the problem should still be considered hard, but not as hard as those problems which admit only fully exponential time algorithms. Formally the running time for the best general algorithms for both of these problems is:

$$O(\exp((c+o(1)))(\ln n)^{1/3}(\ln \ln n)^{2/3})$$

for a constant c.

On the other hand, the best general algorithm for the elliptic curve discrete logarithm problem is fully exponential time - its running time is:

$$O(\sqrt{p}).$$

In simple terms, this means that the elliptic curve discrete logarithm problem is currently considered harder than either the integer factorization problem or the discrete logarithm problem modulo *p*.

As a concrete example, Figure 1 compares the time required to break the ECC with the time required to break RSA or DSA for various modulus sizes using the best general algorithm known. The values are computed in **MIPS years**. A MIPS year represents a computing time of one year on a machine capable of performing one million instructions per second. As a benchmark, it is generally accepted that 10^{12} MIPS years represents reasonable security at this time, since this would require most of the computing power on the planet to work for a considerable amount of time.

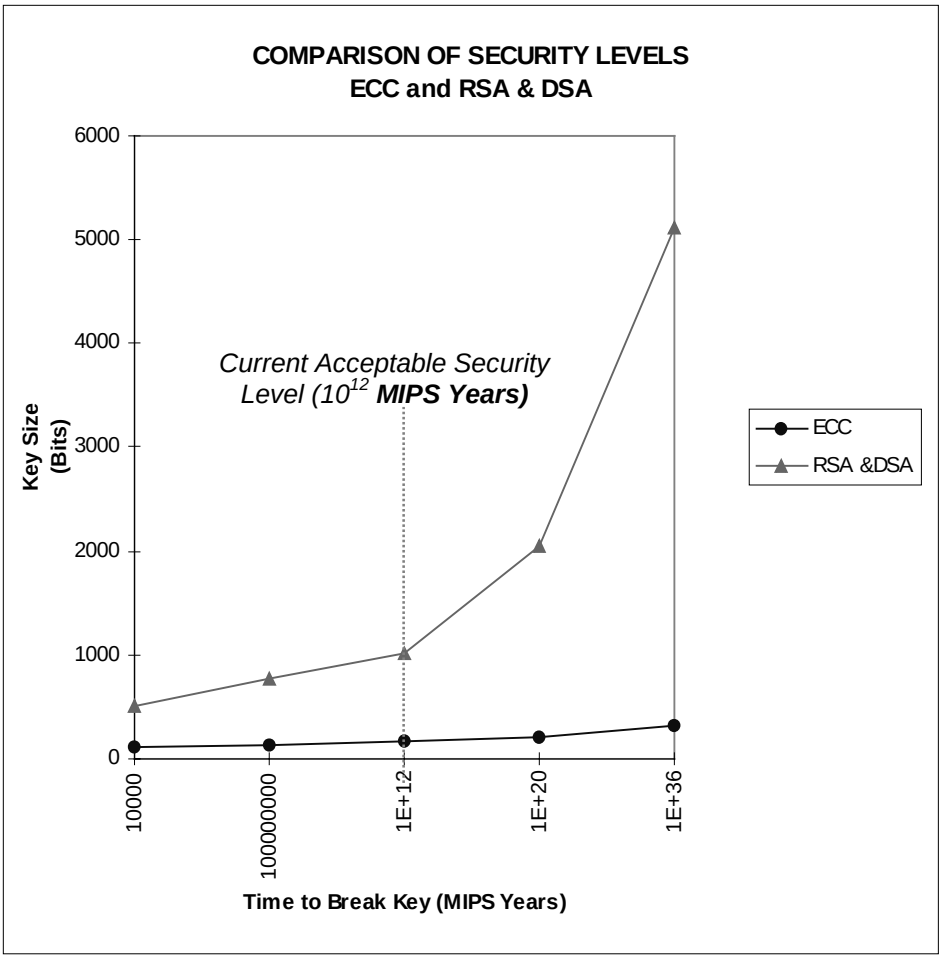

Figure 1: Comparison of Security Levels

Therefore, from the figure, we see that to achieve reasonable security, RSA and DSA should employ a 1024-bit modulus, while a 160-bit modulus should be sufficient for the ECC. Not only can it been seen that the ECC requires a much smaller modulus than RSA or DSA, but also that the security gap between the

systems grows as the key size increases. For example, 300-bit ECC is a great deal more secure than 2000 bit RSA or DSA.

5.2 Efficiency

When talking about the efficiency of a public-key cryptographic system, there are three distinct factors to take into account:

- **computational overheads** - how much computation is required to perform the public key and private key transformations.
- **key size** - how many bits are required to store the key pairs and any system parameters.
- **bandwidth** - how many bits must be communicated to transfer an encrypted message or a signature.

Clearly the comparisons should be made between systems offering similar levels of security, so in order to make the comparisons as concrete as possible, 160-bit ECC is compared with 1024-bit RSA and DSA. As indicated in Section 5.1, these parameter sizes offer comparable levels of security.

Computational Overheads

In each of the systems, considerable computational savings can be made. In RSA, a short public exponent can be employed (although this does incur some security risks) to speed up signature verification and encryption. In both DSA and ECC, a large proportion of the signature generation and encrypting transformations can be pre-computed. Also, various special bases for the finite field F_{2^m} can be employed to perform more quickly the modular arithmetic involved in ECC operation. State-of-the-art implementations of the systems show that with all of these efficiencies in place, ECC is an order of magnitude (roughly 10 times) faster than either RSA or DSA. The use of a short public exponent in RSA can make RSA encryption and signature verification timings (but not RSA decryption and signature generation timings) comparable with timings for these processes using the ECC.

Key Size

Figure 2 on the next page compares the size of the system parameters and key pairs for the different systems.

	System parameters (bits)	Public key (bits)	Private key (bits)
RSA	n/a	1088	2048
DSA	2208	1024	160
ECC	481	161	160

Figure 2: Size of system parameters and key pairs (approx.)

It is clear from the figure that the system parameters and key pairs are shorter for the ECC than for either RSA or DSA.

Bandwidth

All three types of systems have similar bandwidth requirements when they are used to encrypt or sign long messages. However, the case when short messages are being transformed deserves particular attention, because public-key cryptographic systems are often employed to transmit short messages – for example to transport session keys for use in a symmetric-key cryptographic system. For the sake of a concrete comparison, suppose that each is being used to sign a 2000-bit message, or to encrypt a 100-bit message. Figures 3 and 4 compare the lengths of the signatures and encrypted messages respectively.

	Signature size (bits)
RSA	1024
DSA	320
ECC	320

Figure 3: Signature sizes on long messages (e.g. 2000-bit)

	Encrypted message (bits)
RSA	1024
ElGamal	2048
ECC	321

Figure 4: Size of encrypted 100-bit messages

Therefore ECC offers considerable bandwidth savings over the other types of public-key cryptographic systems when being used to transform short messages.

In summary, the ECC provides greater efficiency than either integer factorization systems or discrete logarithm systems, in terms of computational overheads, key sizes, and bandwidth. In implementations, these savings mean higher speeds, lower power consumption, and code size reductions.

6. ECC Standards

In Section 5, this paper discussed the two main issues, namely security and efficiency, that are used to compare public-key cryptographic systems. Other issues that affect the widespread employment of a cryptographic system are interoperability, public acceptance, and technical scrutiny. ECC and other cryptographic systems address these issues through standardization.

The international standardization of cryptographic systems, protocols, and interfaces is an important process and is actively supported by Certicom. Standardization has three main benefits. First, it allows for interoperability among hardware and software systems from many different vendors. Second, it involves critical review of the security of the systems from a cryptographic standpoint. Finally, it permits input into the design of cryptosystems from those who have to implement them in a wide range of environments. Elliptic Curves have been the subject of intensive scrutiny in the mathematical community for many years and have now been scrutinized in standards organizations for over three years. This has given implementors the high degree of confidence in their security that could not be achieved through the support of only a few organizations.

The evolution of standards is a crucial part of the adoption of any cryptographic system. ECC standardization has encouraged its adoption by organizations worldwide. In addition, it has promoted the education of many cryptographers, developers, and engineers in the mathematical basis of ECC and in its importance in achieving practical, efficient public-key based systems.

The following is a list of some ECC standards that have been officially approved or ECC initiatives that are currently underway:

ANSI X9 – The ECC is being incorporated into the American National Standards Institute (ANSI) ASC X9 (Financial Services). ANSI X9.62, “The Elliptic Curve Digital Signature Algorithm” (ECDSA) [7], adopted as an

official ANSI standard in January 1999, aims to achieve a high level of security and interoperability. ANSI X9.63, “Elliptic Curve Key Agreement and Key Management” [8], is currently being drafted, and is expected to be passed in 2000.

ATM Forum – Published in February, 1999, the ATM Forum Technical Committee’s ATM Security Specification Version 1.0 provides security mechanisms for ATM (Asynchronous Transfer Mode) networks. Security services provided include confidentiality, authentication, data integrity, and access control. The ECC is one of the systems supported.

FIPS – The US government's National Institute of Standards and Technology (NIST) announced in February 2000 the extension of its Digital Signature Standard (DSS) to include the ECDSA as specified in ANSI X9.62. The revised standard is FIPS 186-2. This standard is a landmark in the commercial acceptance of ECC since government agencies are now able to purchase security products containing ECC without having to receive special approval. NIST is also including specifications for ECC in its Minimum Interoperability Specification (MISPC).

IEEE P1363 – This project was formally approved as an IEEE standard in February 2000. ECC is included in the standard (Standard Specifications for Public Key Cryptography) [12], which includes encryption, signature, and key agreement mechanisms. Elliptic curves may be defined either modulo p or over F_{2^m}, the field with 2^m elements, for conformance with the standard.

IETF - The OAKLEY Key Determination Protocol of the Internet Engineering Task Force (IETF) describes a key agreement protocol that is a variant of the Diffie-Hellman protocol. It allows for a variety of groups to be used, including elliptic curves. The document makes specific mention of elliptic curve groups over the fields $F_{2^{155}}$ and $F_{2^{210}}$. A draft is available at: http://www.ietf.org/.

ISO/IEC - ECC is being incorporated into several ISO/IEC drafts, in particular, ISO/IEC 14888: "Digital Signature with Appendix Part 3: Certificate-based Mechanisms", ISO/IEC 9796-4: "Digital Signature with Message Recovery, Discrete Logarithm-based Mechanisms", and ISO/IEC 14946: "Cryptographic Techniques Based on Elliptic Curves".

SECG – The Standards for Efficient Cryptography Group is a consortium of leading providers of cryptography and information security solutions who have united to address the lack of interoperability between today’s different cryptographic solutions. The ongoing work of SECG can be found at htttp://www.secg.org/.

WAP (Wireless Application Protocol) - Version 1.0 (released May 1998) provides secure Internet access and other advanced services to digital cellular phones and wireless terminals. ECC is incorporated into the WAP security layer through wTLS (Wireless Transport Layer Security).

In addition to the initiatives underway to draft standards for cryptosystems, numerous initiatives are underway for protocols that use public-key cryptography, public-key certificates, and other public-key management systems. Most of these standards are being written in an algorithm independent manner so that any recognized public-key algorithm can be implemented. This will allow the utilization of algorithms, such as ECC, in environments where other public-key systems would be impractical.

7. ECC Applications

The first whitepaper in this series, "*An Introduction to Information Security*", [9], described a number of applications of public-key cryptographic systems. Some of these applications have already reached the mass market, while others are likely to reach consumers in the next couple of years. The applications discussed included: Automatic Teller Machines (ATMs), stored value phone cards, cellular phones, remote system access, storage of medical records, and electronic cash.

The potential applications for public-key cryptographic systems are endless. In Section 5, we saw that ECC affords more efficient implementations than other public-key systems due to its extra strength. This extra efficiency manifests itself in many ways, including:

- storage efficiencies
- bandwidth savings
- computational efficiencies

These factors, in turn, lead to higher speeds, lower power consumption, and code size reductions.

Therefore, an implementation of ECC is particularly beneficial in applications where bandwidth, processing capacity, power availability, or storage are constrained. Such applications include wireless transactions, handheld computing, broadcast, and smart card applications.

For example, employment of ECC is extremely advantageous in the following situations:

Applications requiring Intensive Public-Key Operations

Many server and financial applications process large volumes of transaction data or Web server requests. Information security through public-key cryptography is required for many electronic operations and is considered a necessity for most Internet-based applications. Even with significant server processing power, server applications can become seriously burdened by

public-key operations and the use of ECC can increase efficiency by orders of magnitude in many cases.

Applications involving Constrained Channels

Constrained channels can be defined as those where computational power is limited at one or both ends, and data transfer speed or bandwidth is limited in the channel between them. The extreme example of this is in wireless communication applications since wireless devices tend to be constrained in computational power, key storage, cryptographic code space, certificate storage, RAM bandwidth, and power. There is a speed/memory tradeoff available with the use of ECC allowing computers that have even modest memory availability to benefit significantly from its use. Through this speed/memory tradeoff, ECC offers benefits in all of these areas. Shorter keys reduce storage space for keys and certificates, and faster computation speeds protocol execution. Power consumption is minimized by efficient processing as well as reducing the amount of data required to be transmitted. This translates into significant power savings in many wireless devices. The ECC offers the further benefit of small code space for these environments.

Use of Smart Cards and Tokens

Cryptographic tokens are portable, physically secure devices used for cryptographic operations. They protect crypto-sensitive operations and store private keys. In order to be practical for widespread distribution, they should also be small, lightweight, and inexpensive. The ideal solution seems to be smart cards (also referred to as chip cards) that can be produced at costs well under $10 and easily transported. Smart cards have extremely rigid constraints on processing power, parameter storage, and code space. They also have fairly slow serial Input/Output, making bandwidth expensive in some cases. They are used primarily for signing and decryption, operations where ECC is ideal. Certicom has done extensive research on card implementations. Certicom will be making its ECC smart card technologies available on a joint development basis so that OEMs can take advantage of more specific implementations.

8. Conclusions

Public-key cryptographic systems have proven to be effective and more manageable than symmetric key systems in a large number of scenarios. Implementors today are faced with a choice between three types of public-key systems: integer factorization systems, discrete logarithm systems, and ECC. Each of these systems is capable of providing confidentiality, authentication, data integrity and nonrepudiation. Of the three, however, the ECC offers significant efficiency savings due to its added strength-per-bit. These savings

are advantageous in many applications, particularly when computational power, bandwidth, or storage space are limited.

9. References and Further Reading

This whitepaper provides a brief introduction to current public-key cryptographic systems, and describes some of the issues involved in public-key cryptography. The following expositions on public-key cryptography in general are recommended as further reading.

[1] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, "Handbook of Applied Cryptography", CRC Press, Boca Raton, Florida, 1997.

[2] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", John Wiley & Sons, New York, 2nd edition, 1996.

[3] D.R. Stinson, "Cryptography: Theory and Practice", CRC Press, Boca Raton, Florida, 1995.

For those wishing to learn more about the ECC, the following texts provide a more comprehensive treatment.

[4] D. Johnson and A.J. Menezes, "Elliptic Curve DSA (ECDSA): An Enhanced DSA", Certicom whitepaper, March 1997.

[5] A.J. Menezes, "Elliptic Curve Public Key Cryptography", Kluwer Academic Publishers, Boston, 1993.

[6] A. Jurisic and A.J. Menezes, "Elliptic curves and cryptography", Dr. Dobb's Journal, pages 26-35, April 1997.

Finally the following references are cited in this whitepaper.

[7] ANSI X9.62, "Public key cryptography for the financial services industry - the Elliptic Curve Digital Signature Algorithm (ECDSA)", January 1999.

[8] ANSI X9.63, "Public key cryptography for the financial services industry - Elliptic Curve Key Agreement and Transport Protocols", draft, 1997.

[9] Certicom Corp., "An Introduction to Information Security", Certicom whitepaper, number 1, March 1997.

[10] W. Diffie and M.E. Hellman, “New directions in cryptography”, IEEE Transactions in Information Theory, volume IT-22, pages 644-654, November 1976.

[11] T. ElGamal, “A public key cryptosystem and a signature scheme based on discrete logarithms”, Advances in Cryptology - Proceedings of CRYPTO’84, Springer Verlag Lecture Notes in Computer Science 196, pages 10-18, 1985.

[12] IEEE P1363, “Standard Specifications for Public-Key Cryptography”, February 2000.

[13] ISO/IEC 14888, “Digital signature with appendix - Part 3: Certificate-based mechanisms”, draft, 1997.

[14] N. Koblitz, “Elliptic curve cryptosystems”, Mathematics of Computation, number 48, pages 203-209, 1987.

[15] A.K. Lenstra, E.R. Verheul, “Selecting Cryptographic Key Sizes”, http://www.cryptosavvy.com

[16] 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.

[17] National Institute of Standards and Technology, “Digital Signature Standard”, FIPS Publication 186, 1993.

[18] M.O. Rabin, “Digitalized signatures and public-key functions as intractable as factorization”, MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[19] 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.

[20] C.P. Schnorr, “Efficient signature generation by smart cards”, Journal of Cryptology, volume 4, pages 161-174, 1991.

[21] H.C. Williams, “A modification of the RSA public-key encryption procedure”, IEEE Transactions on Information Theory, volume IT- 26, pages 726-729, 1980.