ECC

The Elliptic Curve Cryptosystem

THE ELLIPTIC CURVE CRYPTOSYSTEM FOR SMART CARDS

The seventh in a series of ECC white papers

A Certicom White Paper
Published: May 1998

Secure applications in smart cards present implementation challenges particular to the platform's memory, bandwidth, and computation constraints. ECC's unique properties make it especially well suited to smart card applications. Elliptic curve cryptosystems (ECC) provide the highest strength per bit of any cryptosystem known today.

Contents

Please send comments and inquires to:

Certicom Corp.
1400 Fashion Island Boulevard, Suite 1050
San Mateo, California 94404-2062
(650) 312-7960 Fax: (650) 312-7969
info@certicom.com
http://www.certicom.com

This paper is one of a series of Certicom papers. You can download these papers from Certicom's Web site at http://www.certicom.com. Please request hard copies by calling or (650) 312-7960 or emailing info@certicom.com.

1. AN INTRODUCTION TO INFORMATION SECURITY

Published: March 1997
Introduces information security services, public key cryptography, digital certificates, and CAs.

2. CURRENT PUBLIC-KEY CRYPTOGRAPHIC SYSTEMS

Published: April 1997
Introduces the three hard mathematical problem on which the three types of public key cryptosystems are based and discusses the implications of their differences for different applications.

3. ELLIPTIC CURVES AND CRYPTOGRAPHY

Published: April 1997
Mathematical discussion of ECC, DSA, and the basis for the security of ECC. Also published in slightly revised form in *Dr. Dobb's Journal.*

4. ELLIPTIC CURVE DSA (ECDSA): AN ENHANCED DSA

Published: August 1997
Compares ECDSA to DSA in detail.

5. REMARKS ON THE SECURITY OF THE ELLIPTIC CURVE CRYPTOSYSTEM

Published: September 1997
Revised: April 1998
Analyzes the three hard mathematical problem on which the three types of public key cryptosystems are based and discusses the implications of their differences.

6. PERFORMANCE DISCUSSION

Published: August 1997
Brief discussion of why Certicom ECC is faster than RSA with benchmark results on two different platforms.

7. THE ELLIPTIC CURVE CRYPTOSYSTEM FOR SMART CARDS

Published: April 1998
Explains why the unique properties of ECC make it especially suitable for smart card applications.

1. Introduction

With the need for information security in today's digital systems both acute and growing, cryptography has become one of their critical components. Cryptographic services are required across a variety of platforms in a wide range of applications such as secure access to private networks, stored value, electronic commerce, and health care. Incorporating these services into solutions presents an ongoing challenge to manufacturers, systems integrators, and service providers because applications must meet the market requirements of mobility, performance, convenience, and cost containment.

This paper focuses on implementing cryptographic services on the smart card platform, explaining how elliptic curve cryptography (ECC) can not only significantly reduce the cost, but also accelerate the deployment of smart cards in next-generation applications. ECC permits reductions in key and certificate size that translate to smaller memory requirements (especially for EEPROM), which represent significant cost savings. Additionally, because of efficient implementation techniques from Certicom, the ECC algorithm does not require the addition of a cryptographic coprocessor to deliver subsecond performance. This means that high-strength public-key cryptosystems with subsecond transactions times can now be offered on conventional, 8-bit, inexpensive smart cards.

2.0 Public-Key Cryptography

2.1 Cryptographic Services

Today's secure smart card solutions generally employ some combination of four different cryptographic services. These services are:

Authentication	Ensures that the card is the entity claimed. Typically provided by digital signatures.
Nonrepudiation	Makes the actions performed by the cardholder in an electronic transaction nonrevocable so that they are legally binding. Typically provided by digital signatures.
Data integrity	Ensures that the information exchanged in an electronic transaction is not alterable without detection. Typically provided by digital signatures.
Confidentiality	Keeps the data involved in an electronic transaction private. Typically provided by encryption.

2.2 Types of Cryptography

The two main types of cryptography are *symmetric-key* and *public-key* cryptography. Both are based on complex mathematical algorithms and are controlled by keys (digital strings). Symmetric-key cryptography schemes require two parties who want to communicate in confidence to share a common, secret key. Each user must trust the other not to divulge the

common key to a third party. These systems encrypt large amounts of data efficiently; however, they pose significant key management problems in networks of more than a very small number of users, and are typically used in conjunction with public-key cryptography.

Public-key cryptography schemes require each party to have a key pair: a private key, which must not be disclosed to another user, and a public key, which may be made available in a public directory. The two keys are related by a hard one-way function, so it is computationally infeasible to determine the private key from the public key. Because the security of the private key is critical to the security of the cryptosystem, the private key is often stored in software with password protection; however, the private key should ideally be stored in a secure hardware token that prevents direct access or tampering.

Public-key systems solve the key management problems associated with symmetric-key encryption; however, even more importantly, public-key cryptography offers the ability to efficiently implement digital signatures.

2.3 Digital Signatures

As the name suggests, *digital signatures* are the electronic equivalent of traditional handwritten signatures. Handwritten signatures provide a security service because the uniqueness of individuals' handwriting makes signatures hard to forge.

Unlike an individual's handwriting, electronic information is easy to duplicate. If electronic signatures were used in the same way as written signatures (simply appending a fixed string to each message that needed to be signed), security could easily be compromised. For example, to forge Bob's signature, all an eavesdropper (customarily referred to as *Eve*) would need is one previous copy of Bob's signature. She could then append it to any message she chooses, and claim that the message had been signed by Bob.

To avoid this exposure, the digital signature process has been developed. In this process, Bob applies his private key, D_{bob}, to the message using a digital signature algorithm to generate a digital signature. He then sends the message along with the digital signature to Alice. Alice checks, or *verifies*, the signature by applying Bob's public key, E_{bob}, to the signature using a digital signature verification algorithm. Since only Bob is in possession of D_{bob}, Alice knows if the signature verification passes, the message was from Bob and that the data has not been changed.

Digital signatures can be used to deliver three of the cryptographic services introduced earlier: authentication, nonrepudiation, and data integrity. These are generally the most critical security services for smart card applications. ECC can be used to generate strong digital signatures with only a small amount of processing power.

2.4 Elliptic Curve Cryptography

Since the invention of public-key cryptography, numerous public-key cryptographic systems have been proposed. Each of these systems relies on a difficult mathematical problem for its security. None has been *proven* to be intractable (difficult to solve in an efficient manner). Rather, they are *believed* to be intractable because years of intensive study by leading mathematicians and computer scientists has failed to yield efficient algorithms for solving them, so that in practice, they remain intractable with current computational technology. The longer it takes to derive the

key with the best known algorithm for a problem, the more secure a public-key cryptosystem based on that problem is.

Today, three types of systems, classified according to the mathematical problem on which they are based, are generally considered both secure and efficient. The systems 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)
- the elliptic curve discrete logarithm systems (also known as *elliptic curve* cryptosystems).

Elliptic curves are mathematical constructs that have been studied by mathematicians since the seventeenth century. In 1985, Neal Koblitz and Victor Miller independently proposed public-key systems using a group of points on an elliptic curve, and elliptic curve cryptography (ECC) was born. Since that time, numerous researchers and developers have spent several years researching the strength of ECC and improving techniques for its implementation. Today it offers those looking for a smaller, faster public-key system a practical and secure technology for even the most constrained environments.

ECC delivers the highest strength per bit of any known public-key system because of the difficulty of the hard problem upon which it is based. This greater difficulty of the hard problem — the elliptic curve discrete logarithm problem (ECDLP) — means that smaller key sizes yield equivalent levels of security. Table 1 compares the key sizes needed for equivalent strength security in ECC with RSA and DSA. Given the best known algorithms to factor integers and compute elliptic curve logarithms, the key sizes are considered to be equivalent strength based on MIPS years needed to recover one key.

Time to break in MIPS years	RSA/DSA key size	ECC key size	RSA/ECC key size ratio
10^{4}	512	106	5 : 1
10^{8}	768	132	6 : 1
10^{11}	1,024	160	7 : 1
10^{20}	2,048	210	10 : 1
10^{78}	21,000	600	35 : 1

Table 1: Key Size
Equivalent Strength Comparison

The difficulty of the problem and the resulting equivalent-strength key sizes lend several direct benefits to smart card implementations. The following section discusses these benefits.

3.0 Smart Cards and ECC

Smart cards are small, portable, tamper-resistant devices providing users with convenient storage and processing capability. Because of their unique form factor, smart cards are proposed for use in a wide variety of applications such as electronic commerce, identification, and health care. For many of these proposed applications, cryptographic services offered by digital signatures would

be required. To be practical for widespread use, however, smart cards also need to be inexpensive.

The smart card is amenable to cryptographic implementations for several reasons. The card contains many security features that enable the protection of sensitive cryptographic data and provide for a secure processing environment. As mentioned earlier in Section 2.2, the protection of the private key is critical; to provide cryptographic services, this key must never be revealed. The smart card protects the private key, and many consider the smart card an ideal cryptographic token.

3.1 Smart Card Implementation Constraints

Implementing public-key cryptography in a smart card application poses numerous challenges. Smart cards present a combination of implementation constraints that other platforms do not: constrained memory and limited computing power are two of them. The majority of the smart cards on the market today have between 128 and 1024 bytes of RAM, 1 and 16 kilobytes of EEPROM, and 6 and 16 kilobytes of ROM with the traditional 8-bit CPU typically clocked at a mere 3.57 megahertz. Any addition to memory or processing capacity increases the cost of each card because both are extremely cost sensitive.

Smart cards are also slow transmitters, so to achieve acceptable application speeds, data elements must be small (to limit the amount of data passed between the card and the terminal). While cryptographic services that are efficient in memory usage and processing power are needed to contain costs, reductions in transmission times are also needed to enhance usability.

3.2 Meeting the Implementation Constraints with ECC

Less EEPROM and Shorter Transmission Times
The strength (difficulty) of the ECDLP algorithm means that strong security is achievable with proportionately smaller key and certificate sizes. The smaller key size in turn means that less EEPROM is required to store keys and certificates and that less data needs to be passed between the card and the application so that transmission times are shorter.

Scalability
As smart card applications require stronger and stronger security (with longer keys), ECC can continue to provide the security with proportionately fewer additional system resources. This means that with ECC, smart cards are capable of providing higher levels of security without increasing their cost.

No Coprocessor
The nature of the actual computations — more specifically, ECC's reduced processing times — also contribute significantly to why ECC meets the smart card platform requirements so well. Other public-key systems involve so much computation that a dedicated hardware device known as a *crypto coprocessor* is required. The crypto coprocessors not only take up precious space — they add about 20 to 30 percent to the cost of the chip, and about three to five dollars to the cost of each card. With ECC, the algorithm can be implemented in available ROM, so no additional hardware is required to perform strong, fast authentication.

On Card Key Generation
As mentioned earlier, the private key in a public-key pair must be kept secret. For true nonrepudiation, the private key must be completely inaccessible to all other parties. In

applications using the other types of public key systems currently in use, cards are personalized (keys are loaded or injected into the cards) in a secure environment to meet this requirement. Because of the complexity of the computation required, generating keys on the card is inefficient and typically impractical.

With ECC, the time needed to generate a key pair is so short that even a device with the very limited computing power of a smart card can generate the key pair, provided a good random number generator is available. This means that the card personalization process can be streamlined for applications in which nonrepudiation is important.

Summarizing, ECC key size advantages afford many benefits for smart cards, and the superior performance offered by Certicom's ECC implementations make applications feasible in low end devices without dedicated crypto hardware.

3.3 ECC Performance Discussion

Certicom has run benchmarks across several different platforms to test the performance of Certicom's ECC implementation relative to RSA implementations. (See Table 2 for a sample of benchmark data.) Signing speeds are consistently many times faster than those of RSA, although the figures vary widely. For example, signing speeds range from 20 to 300 times faster on mainstream platforms. To account for this discrepancy, note that Certicom's tests included ECC implementations only from Certicom, which incorporate numerous performance-enhancing techniques. Certicom continues to add performance enhancements to its implementations.

Function	Security Builder 1.2 163-bit ECC (ms)	BSAFE 3.0 1,024-bit RSA (ms)
Key Pair Generation	3.8	4,708.3
Sign	2.1 (ECNRA) 3.0 (ECDSA)	228.4
Verify	9.9 (ECNRA) 10.7 (ECDSA)	12.7
Diffie-Hellman Key Exchange	7.3	1,654.0

Table 2: Benchmarks for Solaris (167-MHz UltraSPARC running Solaris 2.5.1)

3.3.1 Background Explanation: Computational Overhead

Public Key	Private Key
Verifying Encrypting	Signing Decrypting

Table 3: Public and Private Key Operations

In any public-key system, the private key is used for signing and decrypting and the public key is used for verifying and encrypting.

The RSA crypto algorithm really only uses one type of calculation — exponentiation. RSA takes a message to the power of an exponent. The nature of the exponent determines the speed of the operation. For signing and decryption, the exponent (private key) is large, so the calculation is quite slow. For verification and encryption, however, the exponent (public key) can be quite small. Some implementers use a public key as small as 3 for very fast verification. Therefore, any analysis of RSA speed consists of slow times for signing and decrypting and much faster speeds for verification and encryption.

In ElGamal systems including DSA and Elliptic Curve DSA (ECDSA), completely different mathematical operations are used for signatures and encryption/decryption, so signing and decryption times do not equal each other, nor do those of verifying and encrypting. And the basic ratio is reversed: Signing is faster than verification; decryption is faster than encryption. More significant is that the difference between public and private key operation times is trivial compared with this difference in RSA. Even though ECC public key operations are a bit slower than those in RSA, the sum of the public and private key times in ECC is much smaller than in RSA.

In both types of systems, considerable computational savings are achievable. In RSA, as previously described, 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 portion of the signature generation and encrypting transformations can be precomputed. Various special bases for the finite field F_{2^m} (see Section 4.0) can be used to accelerate the arithmetic involved in ECC operation.

As mentioned previously, cryptosensitive operations (those using the private key) must be performed in a secure environment because disclosure of the private key would compromise the system. Operations using the public key can often be performed in a nonsecure environment because key disclosure is not an issue.

Since the cryptosensitive operations (signing and decrypting) can be many times faster using ECC than using RSA, ECC is more appropriate for use in secure devices such as smart cards and wireless devices with constrained computational power. The noncrypto-sensitive (public key) operations can usually be performed in terminal or PC environments that typically have more computational power. Because the RSA crypto-sensitive operations require more computational power, they are less suitable for use in constrained environments, and as security (key size) requirements increase in the future, the problem could become worse.

4.0 ECC Implementation Choices

ECC requires the use of two types of mathematics:

- elliptic curve point arithmetic
- the underlying finite field arithmetic.

The developer or implementer must select the underlying *finite field* when implementing ECC. (An *elliptic curve* is a set of points specified by two variables that are elements over a field F_q. A *field* is a set of elements with two custom-defined arithmetic operations, usually addition and multiplication.) Most of the computation for ECC takes place at the finite field level.

The two most common choices for the underlying finite field are:

- F_{2^m}, also known as *characteristic two* or *even* (containing 2^m elements, where m is an integer greater than one)
- F_p, also known as *integers modulo p*, *odd*, or *odd prime* (containing p elements, where p is an odd prime number).

Both of these finite fields are included in draft standards for ECC; mathematical discoveries to date suggest that the ECDLP is equally difficult for instances that use F_{2^m} as for those that use F_p where the sizes of the fields (2^m and p) are approximately equal. Figure 1 illustrates the possible choices for selecting the underlying field.

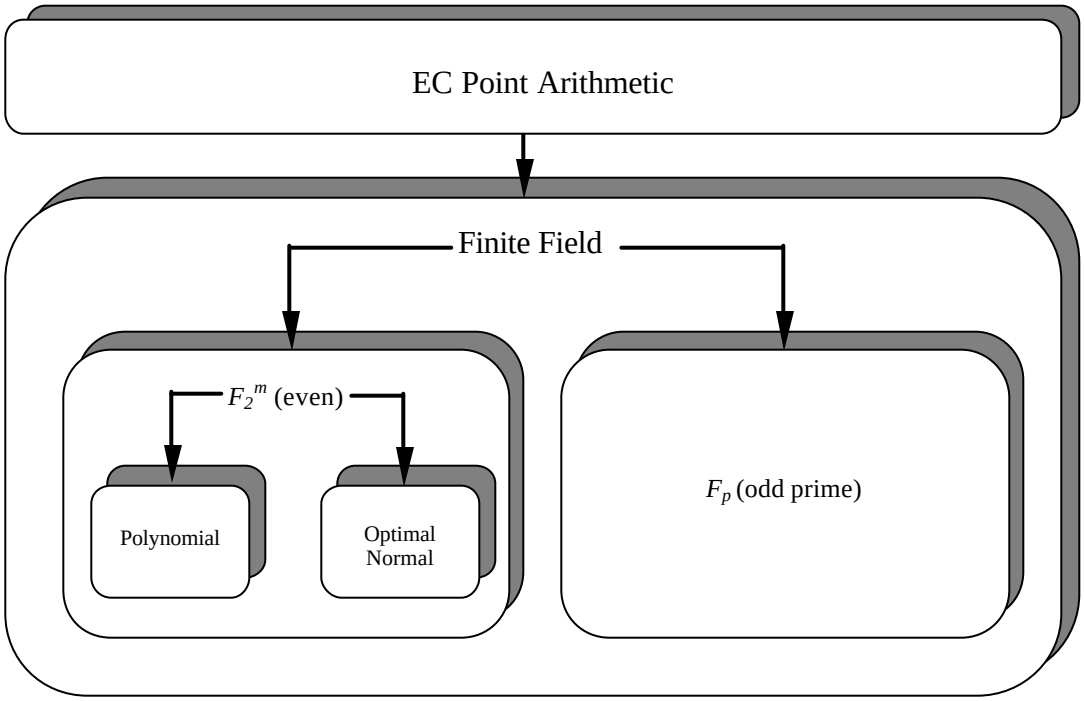

Figure 1: Finite fields over which an elliptic curve can be defined

Although no security or standardization differences exist between the two types of underlying finite fields, performance and cost differences can arise when implementing a smart card application.

In general software environments, the use of F_{2^m} offers significant performance advantages over F_p. This holds true for platforms such as a Sun SPARCstation, an HP server, an embedded system, and more importantly, for a low-cost, 8-bit smart card. To achieve subsecond performance with F_p, a crypto coprocessor (a dedicated hardware component for cryptographic processing) is required. As mentioned in Section 3.2, this additional crypto coprocessor increases the cost of each chip by 20 to 30 percent, which adds three to five dollars to the cost of the card. *With F_{2^m}, a smart card is less expensive because a coprocessor is not needed to deliver subsecond performance.*

In software environments in which an arithmetic processor is already available for modular exponentiation, the performance of F_p can be improved so that in some cases it exceeds the performance of F_{2^m}. This holds true for platforms such as those using Pentium processors or, in the case of smart cards, those having a crypto coprocessor to accelerate modular arithmetic. *If a crypto coprocessor were already available on the smart card (or if the cost associated with adding one were not an issue), then F_p would offer performance advantages over F_{2^m}* implemented without a dedicated hardware component.

Point compression allows the points on an elliptic curve to be represented with fewer bits of data. *In smart card implementations, point compression is essential* because it reduces not only the storage space for keys and certificates on the card, but also the amount of data that needs to be transmitted to and from the card. It can be accomplished with negligible computation using F_{2^m}, but can affect F_p implementations considerably.

F_{2^m} hardware implementations offer significant performance and die size advantages over F_p hardware implementations. Smart cards requiring a number of different cryptographic services with extremely fast performance would require cryptographic coprocessors (unless a high end smart card IC such as a 32-bit device were available). Existing crypto coprocessors, which are optimized for modular arithmetic over F_p, do not substantially increase the performance of F_{2^m} modular arithmetic. A coprocessor designed to optimize F_{2^m} would take up less space on the smart card (thereby reducing the card cost) and would provide superior performance to an F_p implementation.

If the field F_{2^m} is used as the underlying finite field, then the elements of F_{2^m} can be represented in two efficient ways (i.e., the arithmetic can be performed in different ways, depending on how the bit strings are interpreted). These two (most efficient) ways are:

- an optimal normal basis representation
- a polynomial basis representation.

Since elements in one representation can be efficiently converted to elements in the other representation by using an appropriate change-of-basis matrix, interoperability between systems using the two different types of field representation can be achieved easily. As with the choice of the underlying field, the choice of representation does not affect the intractability of the ECDLP.

5.0 Conclusions

Smart cards have extremely rigid constraints on processing power, parameter storage, and code space, as well as slow input/output. As a result, implementation of public-key cryptosystems in smart cards has usually been associated with high-end cards, typically with both large memory configurations and a cryptographic coprocessor.

Certicom's ECC implementations enable the deployment of lower-cost smart cards without compromising any of the required performance and security features. ECC smart cards would not require as much memory, nor would they require a cryptographic coprocessor to deliver strong authentication. Certicom can work with any manufacturer to deliver an ECC-enabled smart card solution. This vendor-independence gives service providers and application developers the flexibility needed to assemble and integrate components for innovative, competitive solutions.

6.0 References and Further Reading

Allen, Catherine A., Barr, William J., and Schultz, Ron, *Smart Cards: Seizing Strategic Business Opportunities: The Smart Card Forum*, Irwin Professional Publications, 1996.

ANSI X9.62, "The elliptic curve digital signature algorithm (ECDSA)," *draft standard*, 1997.

ANSI X9.63, "Elliptic curve key agreement and transport protocols," *draft standard*, 1997.

Frey, G. and Rück, H., "A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves," *Mathematics of Computation*, Volume 62, pages 865-874, 1994.

Harper, G., Menezes, A., and Vanstone, S., "Public-key cryptosystems with very small key lengths," *Advances in Cryptology, EUROCRYPT '92 - Lecture Notes in Computer Science*, Volume 658, Springer-Verlag, pages 163-173, 1993.

Koblitz, N., "Elliptic curve cryptosystems," *Mathematics of Computation*, Volume 48, pages 203-209, 1987.

Koblitz, N., "CM-curves with good cryptographic properties," *Advances in Cryptology, CRYPTO '91 - Lecture Notes in Computer Science,* Volume 576, Springer-Verlag, pages 279-287, 1992.

Koblitz, N., *A Course in Number Theory and Cryptography*, Springer-Verlag, Second Edition, 1994.

Menezes, A., *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers, 1993.

Menezes, A., van Oorschot, P., and Vanstone, S., *Handbook of Applied Cryptography*, CRC Press, 1997.

Menezes, A., Okamoto, T., and Vanstone, S., "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, Volume 39, pages 1639-1646, 1993.

Miller, V., "Uses of elliptic curves in cryptography," *Advances in Cryptology, CRYPTO '85 - Lecture Notes in Computer Science*, Volume 218, Springer-Verlag, pages 417-426, 1986.

Monk, J. Thomas and Dreifus, Henry N., *Smart Cards: A Guide to Building and Managing Smart Card Applications*, John Wiley & Sons, 1997.

van Oorschot, P. and Wiener, M., "Parallel collision search with cryptanalytic applications," in *Proceedings of the Second ACM Conference on Computer and Communications Security*, ACM Press, pages 210-218, 1994.

Pohlig, S. and Hellman, M., "An improved algorithm for computing logarithms over GF(p) and its cryptographic significance," *IEEE Transactions on Information Theory*, Volume 24, pages 106-110, 1978.

Pollard, J., "Monte Carlo methods for index computation mod p," *Mathematics of Computation*, Volume 32, pages 918-924, 1978.

Satoh, T. and Arako, K., "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves," preprint, 1997.

Solinas, J., "An improved algorithm for arithmetic on a family of elliptic curves," *Advances in Cryptology - CRYPTO '97, Lecture Notes in Computer Science*, Volume 1294, Springer-Verlag, pages 357-371, 1997.

Smart, N., "The Discrete Logarithm Problem on Elliptic Curves of Trace One," paper, Hewlett-Packard Laboratories, nsma@hplb.hpl.hp.com,1997.

Zoreda, Jose Luis and Oton, Jose Manuel, *Smart Cards,* Artech House, 1994.