EXHIBIT “A”

Memorandum

To: P1363 working group
From: Alfred Menezes
Subject: IEEE P1363, *Part 6: Elliptic Curve Systems*
Date: October 30, 1994.

Enclosed is a copy of IEEE P1363, Part 6, October 30, 1994, for your review.

I reveived comments from Roger Schlafly and Burt Kaliski. The following is a list of major changes made to the August 19, 1994 draft.

1. Extended the glossary.
2. The signature scheme with appendix was modified.
3. A signature scheme with message recovery was added.
4. Added the section on key lengths.
5. Extended the section of key generation considerations.
6. Added a brief introduction to normal bases.
7. Added a subsection on selecting appropriate curves.
8. Added a subsection on computing the order of a point.
9. Added a list of references.

The following items will be addressed in future drafts of this standard.

1. Complete a detailed described of the ECSSA and ECSSM signature schemes.
2. Use ASN.1 to describe the key syntax.
3. Specify a hash function for use with the ECSSA signature scheme.

I would very much appreciate any editorial suggestions or technical comments.

WORKING DRAFT

IEEE P1363 STANDARD

STANDARD FOR RSA, DIFFIE-HELLMAN AND RELATED PUBLIC-KEY CRYPTOGRAPHY

PART 6: ELLIPTIC CURVE SYSTEMS (Draft 2)

Notice Warning to readers of this document

This document is in the working document stage. It has not yet been processed through the consensus procedures of the IEEE.

Many changes which may greatly affect the contents can occur before this document becomes an IEEE Standard. The developmental committee may not be held responsible for the contents of this document as it currently exists.

Implementation or design based on this working paper is at the risk of the user. No advertisement implying compliance with this "Standard" should appear as it is erroneous and misleading to so state.

Dr. Alfred J. Menezes
Dr. Minghua Qu
Dr. Scott A. Vanstone

Möbius Encryption Technologies
200 Matheson Boulevard, West
Mississauga, Ontario,
Canada, L5R 3L7

October 30, 1994

Outline

Part 6

Elliptic Curve Systems

Abstract. *This standard describes a method for data encryption and for digital signatures using the elliptic curve analogue of the ElGamal public-key cryptosystem. Elliptic curve systems are public-key (asymmetric) cryptographic algorithms, typically used in conjunction with a hash algorithm to create digital signatures, and for the secure distribution of secret keys for use in symmetric cryptosystems. Elliptic curve systems may also be used to transmit confidential information.*

Introduction

The algebraic system defined on the points of an elliptic curve provides an alternate means to implement the ElGamal and ElGamal-like public key encryption and signature protocols. These protocols are typically described in the literature in the algebraic system $\mathbb{Z}_p$, the integers modulo p, where p is a prime. For example, the NIST DSS is an ElGamal-like signature scheme defined over $\mathbb{Z}_p$. Precisely the same protocol for signing could be defined over the points on an elliptic curve.

Elliptic curve systems as applied to ElGamal protocols were first proposed in 1985 independently by Neil Koblitz from the University of Washington, and Victor Miller, who was then at IBM, Yorktown Heights. Elliptic curves as algebraic geometric entities have been studied extensively for the past 150 years, and from these studies has emerged a rich and deep theory. The security of the cryptosystems using elliptic curves hinges on the intractability of the discrete logarithm problem in the algebraic system. It appears to be much more difficult to compute logarithms in an elliptic curve system than to compute logarithms in $\mathbb{Z}_p$. Over the past nine years this problem has received considerable attention from leading mathematicians around the world. No substantial improvements in the ability to find logarithms in an elliptic curve system have been found.

Implementations of elliptic curve cryptosystems offer substantial improvements over existing public key systems including much higher speed, lower power consumption and a smaller key size relative to cryptographic strength. The shorter key size has unique advantages for signing short messages such as those used in electronic funds transfers, cellular and broadcast systems.

Elliptic curves systems have been implemented by various groups around the world including Siemens (Germany), Matsushita (Japan), Thompson (France) and Mobius En-

cryption Technologies (Canada). An ISO IEC SC27 standard for elliptic curve systems is currently being drafted.

Symbols and Notation

$\lceil x \rceil$	The smallest integer $\geq x$. For example, $\lceil 5 \rceil$ 5 and $\lceil 5.3 \rceil$ 6.
$\lfloor x \rfloor$	The largest integer $\leq x$. For example, $\lfloor 5 \rfloor$ 5 and $\lfloor 5.3 \rfloor$ 5.
binary string	A *binary string* is a sequence of 0 s and 1 s. The leftmost bit is the *most significant bit* of the string. The rightmost bit is the *least significant bit* of the string.
$X \oplus Y$	Bitwise *exclusive or* of two binary strings X and Y.
octet	An *octet* is a binary string of length 8. An octet is represented by a hexadecimal string of length 2. The first hexadecimal digit represents the four most significant bits of the octet. The second hexadecimal digit represents the four least significant bits of the octet. For example, 9d represents the binary string 10011101.
octet string	An *octet string* is a sequence of octets.
$X \parallel Y$	Concatenation of two strings X and Y. X and Y are either both binary strings, or both octet strings.
$\|X\|$	Length in octets of the octet string X.
PS	Padding string.
$\log_2 x$	The logarithmic function to the base 2.
$a \bmod n$	The unique remainder r, $0 \leq r \leq n-1$, when integer a is divided by n. For example, 23 mod 7 2.
$\mathbb{Z}_p$ or $\mathbb{F}_p$	The integers modulo p, where p is a prime number.
$\mathbb{F}_{2^m}$	The finite field containing 2^m elements.
$\mathbb{F}_q$	The finite field containing q elements. For this standard, q will either be a prime number (p) or a power of 2 (2^m).
t	A field element of $\mathbb{F}_q$ will be represented as a binary string of length $t \lceil \log_2 q \rceil$. In particular, if $q 2^m$, then a field element in $\mathbb{F}_{2^m}$ can be represented as a binary string of length $t m$.
E	An *elliptic curve* E is specified by 2 parameters a and b, which are elements of a field $\mathbb{F}_q$. The elliptic curve is said to be *defined over* $\mathbb{F}_q$, and $\mathbb{F}_q$ is sometimes called the *underlying field*. If q is a prime (so the field is $\mathbb{F}_p$), then the equation defining the curve is of the form $y^2 x^3 ax b$, where $4a^3 27b^2 / 0$. If q is a power of 2 (so the field is $\mathbb{F}_{2^m}$), then the equation defining the curve is of the form $y^2 xy x^3 ax^2 b$, where $b / 0$.
$\mathcal{O}$	A special point on an elliptic curve, called the *point at infinity*.

P	P is a point (x_P, y_P) on an elliptic curve defined over a field $\mathbb{F}_q$, where x_P and y_P are elements of $\mathbb{F}_q$. The values x x_P and y y_P must satisfy the equation defining E. x_P is called the *x-coordinate* of P and y_P is called the *y-coordinate* of P. There is an addition rule which allows the addition of two elliptic curve points P_1 and P_2 to produce a third elliptic curve point P_3. If k is a positive integer, then kP denotes the point obtained by adding together k copies of the point P.
$\widetilde{y_P}$	Let P be a point (x_P, y_P) on an elliptic curve E defined over a field $\mathbb{F}_q$. If q is a prime, then $\widetilde{y_P}$ is equal to the least significant bit of y_P. If q is a power of 2, then $\widetilde{y_P}$ is 0 if x_P 0. If x_P / 0, then $\widetilde{y_P}$ is equal to the least significant bit of the field element $y_P \cdot {x_P}^{-1}$.
$E(\mathbb{F}_q)$	If E is defined over $\mathbb{F}_q$, then $E(\mathbb{F}_q)$ denotes the number of points on the curve. $E(\mathbb{F}_q)$ is called the *order* of E.
supersingular	An elliptic curve defined over $\mathbb{F}_p$ is *supersingular* if $E(\mathbb{F}_p)$ p 1. An elliptic curve defined over $\mathbb{F}_{2^m}$ is *supersingular* if $E(\mathbb{F}_{2^m})$ is odd.
non-supersingular	If the curve is not supersingular, it is called *non-supersingular.*
n, h	The *order* of the point P is n this is the smallest positive integer such that nP $\mathcal{O}$. The integers in the range $0, n-1$ are represented by binary strings of length h $\lceil \log_2 n \rceil$.
ECES	Elliptic Curve Encryption Scheme.
ECSSA	Elliptic Curve Signature Scheme with Appendix.
ECSSM	Elliptic Curve Signature Scheme with Message Recovery.
ECSS	This refers to either ECSSA or ECSSM.
SHA	The Secure Hash Algorithm. When a message of length less than 2^{64} bits is input, the SHA produces a 160-bit representation of the message called the *message digest* or *hash value*. Any change of the message will, with very high probability, result in a di erent message digest.
ISO	International Organization for Standardization.
IEC	International Electrotechnical Commission.
ANSI	American National Standards Institute.
ASN.1	Abstract Syntax Notation One. A notation for describing abstract types and values, that is described in standard ISO IEC 8824.
BER	Basic Encoding Rules. A set of rules for representing or encoding the values of each ASN.1 type as a string of octets. There is usually more than one way to encode a given value using BER encoding rules. BER is defined in standard ISO IEC 8825.
DER	Distinguished Encoding Rules. A subset of BER, which gives a unique way to represent any ASN.1 value as an octet string. DER is defined in standard ISO IEC 8825.

6.1 Basic Algorithms

This section gives a high-level overview of the elliptic curve encryption scheme (ECES) and two elliptic curve signature schemes (ECSSA and ECSSM).

The cryptosystems are described using an arbitrary elliptic curve E over an arbitrary finite field $\mathbb{F}_q$. Complete details and refinements are provided in Sections 6.2 6.6.

Elliptic Curve Encryption Scheme ECES

System Setup

An underlying finite field $\mathbb{F}_q$ is chosen. An elliptic curve E defined over $\mathbb{F}_q$, and a point P on E are chosen. The order of the point P is denoted by n.
The field $\mathbb{F}_q$, curve E, point P, and order n, comprise the system parameters, and are public information.

Key Generation

Each entity shall perform the following operations.

1. Select a random integer d in the range $1, n-1$.
2. Compute the point Q : dP.
3. The entity s public key consists of the point Q.
4. The entity s private key is the integer d.

Encryption Process

(Entity B sends a message M to entity A)

Entity B performs the following steps:

1. Look up A s public key: Q.
2. Represent the message M as a pair of field elements (m_1, m_2), $m_1 \in \mathbb{F}_q$, $m_2 \in \mathbb{F}_q$.
3. Select a random integer k in the range $1, n-1$.
4. Compute the point (x_1, y_1) : kP.
5. Compute the point (x_2, y_2) : kQ.
6. Combine the field elements m_1, m_2, x_2 and y_2 in a predetermined manner to obtain two field elements c_1 and c_2.
7. Transmit the data c : (x_1, y_1, c_1, c_2) to A.

Decryption Process

(Entity A decrypts ciphertext c (x_1, y_1, c_1, c_2) received from B)

Entity A performs the following steps:

1. Compute the point (x_2, y_2) : $d(x_1, y_1)$, using its private key d.
2. Recover the message m_1 and m_2 from c_1, c_2, x_2 and y_2.

Notes

(a) A technique for representing a message M as a pair of field elements is specified in Section 6.3.

(b) A simple technique for combining m_1, m_2, x_2 and y_2 to obtain c_1 and c_2 is specified in Section 6.3.

(c) An option available is that all entities use the same underlying field $\mathbb{F}_q$, but each entity selects its own elliptic curve E and point P. In this case a description of E and the point P must be included as part of the public key, and hence the public key is longer.

(d) An elliptic curve point P can be specified by its x-coordinate and the bit $\widetilde{y_P}$. The full y-coordinate can then be recovered from this information. Representing a point in this way reduces the length of the public key. For more details of this technique, see Section C.9.

Elliptic Curve Signature Schemes ECSSA and ECSSM

Two signature schemes are described in this standard.

The first scheme ECSSA is used in text hashing mode, and is an example of a *signature scheme with appendix*. In this scheme the message is hashed to a message digest of fixed length, and then this digest is signed. Verification of the signature requires both the signature and the original message.

The second scheme ECSSM is a *signature scheme with message recovery*. In this scheme the message is signed directly, and then the original message can be reconstructed from the signature itself. To guard against forgeries it is important that the message include some pre-specified redundancy. The advantages of a signature scheme with message recovery is that it permits applications without a hash function, and furthermore results in smaller bandwidth for signatures of small messages.

System Setup

This is the same as in Section 6.1.1.

Key Generation

This is the same as in Section 6.1.1.

Signature Generation for ECSSA

(Entity A signs a message M for entity B)

A performs the following steps:

1. Represent the message M as a binary string.

2. Use a hash algorithm to compute the hash value m : $H(M)$.
3. Select a random integer k in the range $1, n-1$.
4. Compute the point (x_1, y_1) : kP.
5. Compute r : $x_1 \bmod n$.
6. Use the private key d to compute s : $k^{-1}(m \quad rd) \bmod n$.
7. Compute $s^{-1} \bmod n$.
8. A sends to B the message M and the signature (r, s^{-1}).

Signature Verification for ECSSA

(Entity B verifies A s signature (r, s^{-1}) for a message M.)

B performs the following steps:

1. Look up A s public key Q.
2. Compute the hash value m : $H(M)$.
3. Compute u : $s^{-1}m \bmod n$ and v : $s^{-1}r \bmod n$.
4. Compute the point (x_2, y_2) : $uP \quad vQ$.
5. Compute r' : $x_2 \bmod n$.
6. Accepts A s signature for message M if and only if $r \quad r'$.

Signature Generation for ECSSM

(Entity A signs a message M for entity B)

A performs the following steps:

1. Represent the message M as a pair of field elements m_1 and m_2, which include some pre-specified redundancy.
2. Select a random integer k in the range $1, n-1$.
3. Compute the point (x_1, y_1) : kP.
4. Compute the field elements r_1 : $m_1 x_1$ and r_2 : $m_2 y_1$.
5. Use the private key d to compute s : $k - d(r_1 \quad r_2) \bmod n$.
6. A sends to B the signature (r_1, r_2, s).

Signature Verification for ECSSM

(Entity B recovers the message and verifies A s signature from (r_1, r_2, s).)

B performs the following steps:

1. Look up A s public key Q.
2. Compute the point (x_2, y_2) : $sP \quad (r_1 \quad r_2)Q$.
3. Compute m_1' : $r_1 x_2^{-1}$ and m_2' : $r_2 y_2^{-1}$ and
4. Accept the signature for the message (m_1', m_2') if and only if (m_1', m_2') contains the pre-specified redundancy.

Notes

(a) A good example of a redundancy generating function is in ISO IEC 9796.

6.2 Services Provided

The cryptographic algorithms described in this standard can be used to provide the following services.

- Privacy (secrecy)
- Entity authentication
- Information authentication
- Digital signatures (non-repudiation)
- Authenticated key exchange

6.3 Encryption

This section describes the ECES encryption process.

The encryption process consists of four steps: encryption-block formatting, elliptic curve computations, message inclusion, and point-to-octet-string conversion.

The input to the encryption process is:

- An octet string M, the message. The length of the message M shall not be more that $2l - 3$ octets l is the length of the field size (q) in octets, that is,

$$l \quad \left\lceil \frac{t}{8} \right\rceil, \quad \text{where } t \quad \lceil \log_2 q \rceil.$$

- Two field elements a and b which describe the elliptic curve equation. A field element is represented by a binary string of length t.
- A field element x_P and the bit $\widetilde{y_P}$, which together describe the point $P \quad (x_P, y_P)$ of order n.
- A field element x_Q and the bit $\widetilde{y_Q}$, which together describe the public-key point $Q \quad (x_Q, y_Q)$.

The output from the encryption process shall be an octet string EM of length $3l \quad 1$, the encrypted message.

Encryption block formatting

1. Pad the message M on the left with a padding string PS of $2l - 3 - \|M\|$ octets, followed by the 00 octet, to form the octet string M':

$$M' \quad \text{PS} \,\|\, 00 \,\|\, M.$$

The octets of the padding string PS should be pseudorandomly generated and non-zero.

2. The string consisting of the first $l-1$ octets of M' is called M_1 and the string consisting of the last $l-1$ octets of M is called M_2:

$$M_1 \quad \text{left half of } M'$$
$$M_2 \quad \text{right half of } M'.$$

Notes

(a) Since t is recommended to be at least 155, the value of l is at least 20, and so the length of the message M can always be up to 37 octets.

(b) Since the padding string PS contains no 00 octets, and the padding string is separated from the message M by a 00 octet, the encryption block can be parsed unambiguously.

(c) The standard may be extended to handle messages of length greater than $2l-3$ octets.

Elliptic curve computations

1. Select a random integer k in the range $1, n-1$.
2. Recover the y-coordinate y_P of the point P from x_P and the bit $\widetilde{y_P}$.
3. Compute the elliptic curve point $(x_1, y_1) : \quad kP$, where P is the point (x_P, y_P).
4. Recover the y-coordinate y_Q of the point Q from x_Q and the bit $\widetilde{y_Q}$.
5. Compute the elliptic curve point $(x_2, y_2) : \quad kQ$, where Q is the point (x_Q, y_Q).

Notes

(a) For reasons of e ciency, the integer k may be chosen by setting j randomly chosen positions of its binary representation to 1, and the remaining positions to 0. For security, the value of j should be at least 30.

Message inclusion

1. Convert M_1 to a binary string, and then append a zero binary string of length $8-8l \quad t$ to the left of this binary string to form a field element m_1.
2. Convert M_2 to a binary string, and then append a zero binary string of length $8-8l \quad t$ to the left of this binary string to form a field element m_2.
3. Form the field element x_3 by setting to 0 the first $8-8l \quad t$ most significant bits of x_2.
4. Form the field element y_3 by setting to 0 the first $8-8l \quad t$ most significant bits of y_2.
5. Form the field element x_4 by concatenating the $\lceil \frac{t}{2} \rceil$ most significant bits of x_3 followed by the $\lfloor \frac{t}{2} \rfloor$ least significant bits of y_3.
6. Form the field element y_4 by concatenating the $\lceil \frac{t}{2} \rceil$ most significant bits of y_3 followed by the $\lfloor \frac{t}{2} \rfloor$ least significant bits of x_3.
7. Compute $z_1 : \quad m_1 \oplus y_3$ and then perform a field multiplication to obtain $c_1 : \quad x_4 \cdot z_1$.
8. Compute $z_2 : \quad m_2 \oplus x_3$ and then perform a field multiplication to obtain $c_2 : \quad y_4 \cdot z_2$.

Notes

(a) The value of $8 - 8l \quad t$ is either 1, 2, 3, 4, 5, 6, 7, or 8.

(b) The most significant bit of the field elements m_1, m_2, x_3 and y_3 is 0. The reason for doing this is to ensure, in the case where q is equal to a prime p, that the integers represented by x_4, y_4, z_1 and z_2 are less than the modulus p.

(c) The reason for combining the field elements m_1, m_2, x_2 and y_2 in the manner specified, is to ensure that an eavesdropper who knows c_1, c_2 and also half the message, say m_1, cannot recover the second half of the message m_2, nor can he substitute m_1 by another message m_1' of his choice.

Point to octet string conversion

1. Append a zero binary string of length $8l - t$ to the left of the field element x_1, and convert the resulting $8l$-bit binary string to an octet string X_1 of length l.
2. Compute the bit $\widetilde{y_1}$. Assign the single octet Y_1 the value 00 if $\widetilde{y_1}$ is 0, or the value 01 if $\widetilde{y_1}$ is 1.
3. Append a zero binary string of length $8l - t$ to the left of the field element c_1, and convert the resulting $8l$-bit binary string to an octet string C_1 of length l.
4. Append a zero binary string of length $8l - t$ to the left of the field element c_2, and convert the resulting $8l$-bit binary string to an octet string C_2 of length l.
5. Finally, obtain the ciphertext EM by concatenating the four octet strings X_1, Y_1, C_1 and C_2:

$$\text{EM} \quad X_1 \| Y_1 \| C_1 \| C_2.$$

EM is $3l \quad 1$ octets in length.

6.4 Decryption

This section describes the ECES decryption process.

The decryption process consists of four steps: octet-string-to-point conversion, elliptic curve computations, message extraction, and encryption-block parsing.

The input to the decryption process is:

- An octet string EM of length $3l \quad 1$, the encrypted message.
- Two field elements a and b which describe the elliptic curve equation.
- An integer d, the private key.

The output from the encryption process shall be an octet string M of length at most $2l - 3$, the plaintext message.

Octet string to point conversion

1. Parse the encrypted message EM to obtain octet strings X_1, Y_1, C_1 and C_2, of length l, 1, l and l, respectively:
$$\text{EM} \quad X_1 \,\|\, Y_1 \,\|\, C_1 \,\|\, C_2.$$
2. Convert X_1 to a binary string, and then discard the $8l - t$ most significant bits to obtain a field element x_1.
3. Set the bit $\widetilde{y_1}$ to be 0 if the octet Y_1 is 00, or 1 if the octet Y_1 is 01.
4. Convert C_1 to a binary string, and then discard the $8l - t$ most significant bits to obtain a field element c_1.
5. Convert C_2 to a binary string, and then discard the $8l - t$ most significant bits to obtain a field element c_2.

Elliptic curve computations

1. Use x_1 and $\widetilde{y_1}$ to obtain the elliptic curve point (x_1, y_1).
2. Compute the elliptic curve point $(x_2, y_2) : \quad d(x_1, y_1)$.

Message extraction

1. Form the field element x_3 by setting to 0 the first $8 - 8l \quad t$ most significant bits of x_2.
2. Form the field element y_3 by setting to 0 the first $8 - 8l \quad t$ most significant bits of y_2.
3. Form the field element x_4 by concatenating the $\lceil \frac{t}{2} \rceil$ most significant bits of x_3 followed by the $\lfloor \frac{t}{2} \rfloor$ least significant bits of y_3.
4. Form the field element y_4 by concatenating the $\lceil \frac{t}{2} \rceil$ most significant bits of y_3 followed by the $\lfloor \frac{t}{2} \rfloor$ least significant bits of x_3.
5. Compute $z_1 : \quad c_1 \cdot x_4^{-1}$ and then $m_1 : \quad z_1 \oplus y_3$.
6. Compute $z_2 : \quad c_2 \cdot y_4^{-1}$ and then $m_2 : \quad z_2 \oplus x_3$.
7. Discard the $8 - 8l \quad t$ most significant bits of m_1, and convert the resulting binary string to an octet string M_1 of length $l - 1$.
8. Discard the $8 - 8l \quad t$ most significant bits of m_2, and convert the resulting binary string to an octet string M_2 of length $l - 1$.

Encryption block parsing

1. Concatenate M_1 and M_2 to obtain an octet string M':
$$M' \quad M_1 \,\|\, M_2.$$
2. Parse M' to obtain the message M:
$$M' \quad \text{PS} \,\|\, 00 \,\|\, M.$$

6.5 Signature

ECSSA

ECSSM

6.6 Signature Verification

ECSSA

ECSSM

6.7 Key Length Considerations

The security of the elliptic curve schemes described in this standard hinges on the apparent di culty of the discrete logarithm problem in elliptic curves. To avoid the best known attacks on the discrete logarithm problem, (see Appendix F) the underlying field $\mathbb{F}_q$, the curve E, and the point P should be selected so that the order n of P is divisible by a prime number r which is at least 10^{44} (and hence, q should also be at least 10^{44}).

One simple way to ensure that this condition is met is to select a curve whose order is prime.

6.8 Key Generation Considerations

This section describes ECES and ECSS key generation.

System Setup

The system parameters, namely the field $\mathbb{F}_q$, the field elements a and b, the point P, and the order n, are selected by the system administrator. The system parameters are public information.

1. An underlying finite field $\mathbb{F}_q$ is chosen. The field is either $\mathbb{F}_p$ (so q p, an odd prime) or $\mathbb{F}_{2^m}$ (so q 2^m). Field elements are represented by binary strings of length t $\lceil \log_2 q \rceil$.
2. If q p, then two elements $a, b \in \mathbb{F}_p$ are selected so that $4a^3$ $27b^2$ / 0 in $\mathbb{F}_p$. The elements a and b define an elliptic curve $E : y^2$ x^3 ax b.
   If q 2^m, then two elements $a, b \in \mathbb{F}_{2^m}$ are selected so that b / 0. The elements a and b define an elliptic curve $E : y^2$ xy x^3 ax^2 b.
   In either case, $E(\mathbb{F}_q)$ should be divisible by a large prime number. Preferably, $E(\mathbb{F}_q)$ should be a prime itself.
3. A point P (x_P, y_P) on the elliptic curve E is selected so that the order of P, denoted n, is a prime number. The bit $\widetilde{y_P}$ is computed. The point P is represented by x_P and $\widetilde{y_P}$.

Notes

(a) See Section 6.7 for a discussion of conditions imposed on the sizes of q and n due to security constraints.

(b) See Section C.7 for a discussion on how to select a curve of appropriate order.

(c) See Section C.8 for a method of computing the order of a point.

(d) See Section C.9 for a method of representing a point by the x-coordinate and only one bit of the y-coordinate.

Key Generation

After system setup, each entity performs the following operations.

1. Select a random integer d in the range $1, n-1$.
2. Compute the point Q : dP.
3. Let Q (x_Q, y_Q), and compute $\widetilde{y_Q}$. The entity s public key consists of the point Q, which is represented by x_Q and $\widetilde{y_Q}$.
4. The entity s private key is the integer d.

6.9 Key Syntax

The section describes the syntax for ECES and ECSS public and private keys.

Public Key Syntax

An ECES or ECSS public key is a binary string

$$\text{ECPublicKey} : \quad x_Q \,\|\, \widetilde{y_Q}$$

where x_Q is the x-coordinate of the public-key point Q, and $\widetilde{y_Q}$ is the bit which can be used to recover the y-coordinate y_Q of Q.

Note that ECPublicKey is a binary string of length t 1.

Private Key Syntax

An ECES or ECSS private key is an integer d in the range $1, n-1$.

6.10 Applications (not part of standard)

Appendix C

Mathematical Background

C.1 The Finite Field $\mathbb{F}_p$

Let p be a prime number. The *finite field* $\mathbb{F}_p$ is comprised of the set of integers $\{0, 1, 2, \ldots, p-1\}$ with the following arithmetic operations:

- *Addition:* If $a, b \in \mathbb{F}_p$, then $a \quad b \quad r$ where r is the remainder when $a \quad b$ is divided by p, $0 \leq r \leq p-1$.
- *Multiplication:* If $a, b \in \mathbb{F}_p$, then $ab \quad s$ where s is the remainder when ab is divided by p, $0 \leq s \leq p-1$.

Let $\mathbb{F}_p{}^*$ denote all the non-zero elements in $\mathbb{F}_p$. In $\mathbb{F}_p$, there exists at least one element g such that any non-zero element of $\mathbb{F}_p$ can be expressed as a power of g. Such an element g is called a *generator* (or *primitive element*) of $\mathbb{F}_p$. That is

$$\mathbb{F}_p{}^* \quad \{g^i : 0 \leq i \leq p-2\}.$$

Example (*The finite field* $\mathbb{F}_2$)
$\mathbb{F}_2 \quad \{0,1\}$. The addition and multiplication tables for $\mathbb{F}_2$ are

	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

Example (*The finite field* $\mathbb{F}_{23}$)
$\mathbb{F}_{23} \quad \{0, 1, 2, \ldots, 22\}$. Examples of the arithmetic operations in $\mathbb{F}_{23}$ are $12 \quad 20 \quad 32 \quad 9$, $8 \cdot 9 \quad 72 \quad 3$. The element 5 is a generator of $\mathbb{F}_{23}^*$. The powers of 5 are:

$5^0 \quad 1$ $5^1 \quad 5$ $5^2 \quad 2$ $5^3 \quad 10$ $5^4 \quad 4$ $5^5 \quad 20$
$5^6 \quad 8$ $5^7 \quad 17$ $5^8 \quad 16$ $5^9 \quad 11$ $5^{10} \quad 9$ $5^{11} \quad 22$
$5^{12} \quad 18$ $5^{13} \quad 21$ $5^{14} \quad 13$ $5^{15} \quad 19$ $5^{16} \quad 3$ $5^{17} \quad 15$
$5^{18} \quad 6$ $5^{19} \quad 7$ $5^{20} \quad 12$ $5^{21} \quad 14$ $5^{22} \quad 1.$

C.2 The Finite Field $\mathbb{F}_{2^m}$

Let $f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_2x^2 + f_1x + f_0$, $f_i \in \mathbb{F}_2$, be an irreducible polynomial of degree m over $\mathbb{F}_2$, i.e., $f(x)$ cannot be factored into two polynomials over $\mathbb{F}_2$ each of degree less than m. The *finite field* $\mathbb{F}_{2^m}$ is comprised of all polynomials over $\mathbb{F}_2$ of degree less than m:

$$\mathbb{F}_{2^m} = \{a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 : a_i \in \{0,1\}\}.$$

The field element $(a_{m-1}x^{m-1} + \cdots + a_1x + a_0)$ is usually denoted by the binary string $(a_{m-1}\cdots a_1a_0)$ of length m, so that

$$\mathbb{F}_{2^m} = \{(a_{m-1}\cdots a_1a_0) \ : \ a_i \in \{0,1\}\}.$$

Thus the elements of $\mathbb{F}_{2^m}$ can be represented by the set of all binary strings of length m.

Field elements are added and multiplied as follows:

- *Field addition:* $(a_{m-1}\cdots a_1a_0) + (b_{m-1}\cdots b_1b_0) = (c_{m-1}\cdots c_1c_0)$, where $c_i = a_i + b_i$ in the field $\mathbb{F}_2$. That is, field addition is performed componentwise.
- *Field multiplication:* $(a_{m-1}\cdots a_1a_0)\cdot(b_{m-1}\cdots b_1b_0) = (r_{m-1}\cdots r_1r_0)$, where the polynomial $(r_{m-1}x^{m-1} + \cdots + r_1x + r_0)$ is the remainder when the polynomial $(a_{m-1}x^{m-1} + \cdots + a_1x + a_0)\cdot(b_{m-1}x^{m-1} + \cdots + b_1x + b_0)$ is divided by $f(x)$ over $\mathbb{F}_2$.

Note that $\mathbb{F}_{2^m}$ contains exactly 2^m elements. Let $\mathbb{F}_{2^m}^*$ denote the set of all non-zero elements in $\mathbb{F}_{2^m}$. There exists at least one element g in $\mathbb{F}_{2^m}$ such that any non-zero element of $\mathbb{F}_{2^m}$ can be expressed as a power of g. Such an element g is called a *generator* (or *primitive element*) of $\mathbb{F}_{2^m}$. That is

$$\mathbb{F}_{2^m}^* = \{g^i : 0 \le i \le 2^m - 2\}$$

Example (*The finite field* $\mathbb{F}_{2^4}$)
Take $f(x) = x^4 + x + 1$ over $\mathbb{F}_2$ it can be verified that $f(x)$ is irreducible over $\mathbb{F}_2$. Then the elements of $\mathbb{F}_{2^4}$ are:

$$\begin{array}{cccccccc} (0000) & (0001) & (0010) & (0011) & (0100) & (0101) & (0110) & (0111) \\ (1000) & (1001) & (1010) & (1011) & (1100) & (1101) & (1110) & (1111). \end{array}$$

As examples of field arithmetic, we have $(1011) + (1001) = (0010)$, and

$$\begin{aligned} (1101)\cdot(1001) &= (x^3 + x^2 + 1)(x^3 + 1) = x^6 + x^5 + x^2 + 1 \\ &= (x^4 + x + 1)(x^2 + x) + (x^3 + x^2 + x + 1) \\ &= x^3 + x^2 + x + 1 \bmod f(x) \\ &= (1111) \end{aligned}$$

i.e., the remainder when $(x^3 + x^2 + 1)(x^3 + 1)$ is divided by $f(x)$ is $x^3 + x^2 + x + 1$.

$\mathbb{F}_{2^4}^*$ can be generated by one element $\alpha = x$. The powers of α are:

$$\begin{array}{cccccccc} \alpha^0 & (0001) & \alpha^1 & (0010) & \alpha^2 & (0100) & \alpha^3 & (1000) \\ \alpha^4 & (0011) & \alpha^5 & (0110) & \alpha^6 & (1100) & \alpha^7 & (1011) \\ \alpha^8 & (0101) & \alpha^9 & (1010) & \alpha^{10} & (0111) & \alpha^{11} & (1110) \\ \alpha^{12} & (1111) & \alpha^{13} & (1101) & \alpha^{14} & (1001) & \alpha^{15} = \alpha^0 & (0001). \end{array}$$

C.3 Elliptic Curves over $\mathbb{F}_p$

Let $p > 3$ be a prime number. Let $a, b \in \mathbb{F}_p$ be such that $4a^3 \quad 27b^2 \ / \ 0$ in $\mathbb{F}_p$. An *elliptic curve* $E(\mathbb{F}_p)$ over $\mathbb{F}_p$ defined by the parameters a and b is the set of solutions (x, y) in $\mathbb{F}_p \times \mathbb{F}_p$ to the equation

$$y^2 \quad x^3 \quad ax \quad b,$$

together with an extra point $\mathcal{O}$, the *point at infinity*. The number of points in $E(\mathbb{F}_p)$ is denoted by $E(\mathbb{F}_p)$. The Hasse Theorem tells us that

$$p \quad 1 - 2\sqrt{p} \leq \quad E(\mathbb{F}_p) \leq p \quad 1 \quad 2\sqrt{p}.$$

The set of points $E(\mathbb{F}_p)$ form a group with respect to the following addition rules:

(i) $\mathcal{O} \quad \mathcal{O} \quad \mathcal{O}$.

(ii) $(x, y) \quad \mathcal{O} \quad (x, y)$ for all $(x, y) \in E(\mathbb{F}_p)$.

(iii) $(x, y) \quad (x, -y) \quad \mathcal{O}$ for all $(x, y) \in E(\mathbb{F}_p)$ (i.e., the inverse of the point (x, y) is the point $(x, -y)$).

(iv) (Rule for adding two distinct points that are not inverses of each other)
Let $(x_1, y_1) \in E(\mathbb{F}_p)$ and $(x_2, y_2) \in E(\mathbb{F}_p)$ be two points. If $x_1 \ / \ x_2$, then (x_1, y_1) $(x_2, y_2) \quad (x_3, y_3)$, where

$$x_3 \quad \lambda^2 - x_1 - x_2, \quad y_3 \quad \lambda(x_1 - x_3) - y_1, \quad \text{and} \quad \lambda \quad \frac{y_2 - y_1}{x_2 - x_1}.$$

(v) (Rule for doubling a point)
Let $(x_1, y_1) \in E(\mathbb{F}_p)$ be a point with $y_1 \ / \ 0$. Then $2(x_1, y_1) \quad (x_3, y_3)$, where

$$x_3 \quad \lambda^2 - 2x_1, \quad y_3 \quad \lambda(x_1 - x_3) - y_1, \quad \text{and} \quad \lambda \quad \frac{3x_1^2 \quad a}{2y_1}.$$

The group $E(\mathbb{F}_p)$ is abelian, which means that $P \quad Q \quad Q \quad P$ for all points P and Q in $E(\mathbb{F}_p)$. The curve is said to be *supersingular* if $E(\mathbb{F}_p) \quad p \quad 1$ otherwise it is *non-supersingular*.

Example (*An elliptic curve over* $\mathbb{F}_{23}$)
Let $y^2 \quad x^3 \quad x \quad 1$ be an equation over $\mathbb{F}_{23}$ (i.e., $a \quad 1$ and $b \quad 1$). Then the solutions over $\mathbb{F}_{23}$ to the equation of the elliptic curve are:

(0, 1)	(0,22)	(1,7)	(1, 16)	(3, 10)	(3,13)	(4, 0)	(5, 4)	(5,19)
(6, 4)	(6,19)	(7,11)	(7,12)	(9, 7)	(9,16)	(11,3)	(11,20)	(12,4)
(12,19)	(13,7)	(13,16)	(17,3)	(17,20)	(18,3)	(18,20)	(19,5)	(19,18).

The group $E(\mathbb{F}_{23})$ has 28 points (including the point at infinity $\mathcal{O}$). The following are examples of the group operation.

1. Let $P_1 = (3, 10)$, $P_2 = (9, 7)$, $P_1 + P_2 = (x_3, y_3)$. Compute

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} = 11 \in \mathbb{F}_{23},$$

$$x_3 = 11^2 - 3 - 9 = 6 - 3 - 9 = -6 = 17,$$

$$y_3 = 11(3-17) - 10 = 11(9) - 10 = 89 = 20.$$

Therefore $P_1 + P_2 = (17, 20)$.

2. Let $P_1 = (3, 10)$, $2P_1 = (x_3, y_3)$. Compute

$$\lambda = \frac{3(3^2) + 1}{20} = \frac{5}{20} = \frac{1}{4} = 6,$$

$$x_3 = 6^2 - 6 = 30 = 7,$$

$$y_3 = 6(3-7) - 10 = -24 - 10 = -11 = 12.$$

Therefore $2P_1 = (7, 12)$.

C.4 Elliptic Curves over $\mathbb{F}_{2^m}$

A non-supersingular *elliptic curve* $E(\mathbb{F}_{2^m})$ over $\mathbb{F}_{2^m}$ defined by the parameters $a, b \in \mathbb{F}_{2^m}$, $b \neq 0$, is the set of solutions (x, y) in $\mathbb{F}_{2^m} \times \mathbb{F}_{2^m}$ to the equation

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

together with an extra point $\mathcal{O}$, the *point at infinity*. The number of points in $E(\mathbb{F}_{2^m})$ is denoted by $\#E(\mathbb{F}_{2^m})$. The Hasse Theorem tells us that

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_{2^m}) \leq q + 1 + 2\sqrt{q},$$

where $q = 2^m$. Furthermore, $\#E(\mathbb{F}_{2^m})$ is even.

The set of points $E(\mathbb{F}_{2^m})$ is a group with respect to the following addition rules:

(i) $\mathcal{O} + \mathcal{O} = \mathcal{O}$.

(ii) $(x, y) + \mathcal{O} = (x, y)$ for all $(x, y) \in E(\mathbb{F}_{2^m})$.

(iii) $(x, y) + (x, x + y) = \mathcal{O}$ for all $(x, y) \in E(\mathbb{F}_{2^m})$ (i.e., the inverse of the point (x, y) is the point $(x, x + y)$).

(iv) (Rule for adding two distinct points that are not inverses of each other)
Let $(x_1, y_1) \in E(\mathbb{F}_{2^m})$ and $(x_2, y_2) \in E(\mathbb{F}_{2^m})$ be two points. If $x_1 \neq x_2$, then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, where

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a, \quad y_3 = \lambda(x_1 + x_3) + x_3 + y_1, \text{ and } \lambda = \frac{y_1 + y_2}{x_1 + x_2}.$$

(v) (Rule for doubling a point)
Let $(x_1, y_1) \in E(\mathbb{F}_{2^m})$ be a point with $x_1 \neq 0$. Then $2(x_1, y_1) = (x_3, y_3)$, where

$$x_3 = x_1^2 + \frac{b}{x_1^2}, \text{ and } y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right) x_3 + x_3.$$

The group $E(\mathbb{F}_{2^m})$ is abelian, which means that $P + Q = Q + P$ for all points P and Q in $E(\mathbb{F}_{2^m})$.

Example (*An elliptic curve over* $\mathbb{F}_{2^4}$.)
Consider the field $\mathbb{F}_{2^4}$ generated by the root $\alpha = x$ of the irreducible polynomial $f(x) = x^4 + x + 1$. Consider the non-supersingular elliptic curve over $\mathbb{F}_{2^4}$ with defining equation

$$y^2 + xy = x^3 + \alpha^4 x^2 + 1$$

(so $a = \alpha^4$, $b = 1$). Then the solutions over $\mathbb{F}_{2^4}$ to the equation of the elliptic curve are:

$$\begin{array}{cccccccc} & (0,1) & (1,\alpha^6) & (1,\alpha^{13}) & (\alpha^3,\alpha^8) & (\alpha^3,\alpha^{13}) & (\alpha^5,\alpha^3) & (\alpha^5,\alpha^{11}) \\ (\alpha^6,\alpha^8) & (\alpha^6,\alpha^{14}) & (\alpha^9,\alpha^{10}) & (\alpha^9,\alpha^{13}) & (\alpha^{10},\alpha^1) & (\alpha^{10},\alpha^8) & (\alpha^{12},0) & (\alpha^{12},\alpha^{12}). \end{array}$$

The group $E(\mathbb{F}_{23})$ has 16 points (including the point at infinity $\mathcal{O}$). The following are examples of the group operation.

1. Let $P_1 = (\alpha^6, \alpha^8)$, $P_2 = (\alpha^3, \alpha^{13})$, and let $P_1 + P_2 = (x_3, y_3)$. Then

$$x_3 = \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)^2 + \frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3} + \alpha^6 + \alpha^3 + \alpha^4 = \left(\frac{\alpha^3}{\alpha^2}\right)^2 + \frac{\alpha^3}{\alpha^2} + \alpha^2 + \alpha^4 = 1,$$

$$y_3 = \left(\frac{\alpha^8 + \alpha^{13}}{\alpha^6 + \alpha^3}\right)(\alpha^6 + 1) + 1 + \alpha^8 = \left(\frac{\alpha^3}{\alpha^2}\right)\alpha^{13} + \alpha^2 + \alpha^{13}.$$

2. If $2P_1 = (x_3, y_3)$, then

$$x_3 = (\alpha^6)^2 + \frac{1}{(\alpha^6)^2} = \alpha^{12} + \alpha^3 = \alpha^{10},$$

$$y_3 = (\alpha^6)^2 + \left(\alpha^6 + \frac{\alpha^8}{\alpha^6}\right)\alpha^{10} + \alpha^{10} = \alpha^3 + (\alpha^6 + \alpha^2)\alpha^{10} = \alpha^8.$$

C.5 Computing the Multiple of a Point

If k is a positive integer and P is an elliptic curve point, then kP is the point obtained by adding together k copies of P. This computation can be performed e ciently by the repeated double-and-add method outlined below.

Input: A positive integer k, and an elliptic curve point P.
Output: The elliptic curve point kP.

1. Let $k \quad k_r k_{r-1} \ldots k_1 k_0$ be the binary representation of k, where the most significant bit k_r of k is 1.
2. Set $Q \longleftarrow P$.
3. For i from $r - 1$ downto 0 do

   3.1 Set $Q \longleftarrow Q \quad Q$.
   3.2 If $k_i \quad 1$ then set $Q \longleftarrow Q \quad P$.

4. Output Q.

There are several variations of this method which can be used to speed up the computations. One such method which requires some precomputations is described in 5 .

C.6 Normal Bases in $\mathbb{F}_{2^m}$

Arithmetic in the finite field $\mathbb{F}_{2^m}$ can be performed e ciently both in hardware and in software when the field elements are represented with respect to a normal basis.

The field $\mathbb{F}_{2^m}$ can be viewed as a vector space of dimension m over $\mathbb{F}_2$. That is, there exists a set of m elements $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ in $\mathbb{F}_{2^m}$ such that each $\alpha \in \mathbb{F}_{2^m}$ can be written uniquely in the form

$$\alpha \qquad \sum_{i=0}^{m-1} a_i \alpha_i, \quad \text{where } a_i \in \{0, 1\}.$$

We can then represent α as the 0-1 vector $(a_0, a_1, \ldots, a_{m-1})$. Addition of field elements is performed by bitwise XOR-ing the vector representations.

In general, there are many di erent bases of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$. A *normal basis* of $\mathbb{F}_{2^m}$ over $\mathbb{F}_2$ is a basis of the form

$$\{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\},$$

where $\beta \in \mathbb{F}_{2^m}$ it is a well-known fact that such a basis always exists. Given any element $\alpha \in \mathbb{F}_{2^m}$, we can write $\alpha \quad \sum_{i=0}^{m-1} a_i \beta^{2^i}$, where $a_i \in \{0, 1\}$. Since squaring is a linear operator in $\mathbb{F}_{2^m}$, we have

$$\alpha^2 \qquad \sum_{i=0}^{m-1} a_i \beta^{2^{i+1}} \qquad \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} \qquad (a_{m-1}, a_0, \ldots, a_{m-2}),$$

with indices reduced modulo m. Hence a normal basis representation of $\mathbb{F}_{2^m}$ is advantageous because squaring a field element can then be accomplished by a simple rotation of the vector representation, an operation that is easily implemented in hardware.

Another important property of normal bases to note is that the multiplicative identity element is represented by the all-ones vector of length m.

Multiplying field elements and computing inverses can be done e ciently but is more complicated to describe. For some pointers to the literature, consult the references on page 29.

C.7 Selecting an Appropriate Curve

There are three approaches to selecting an elliptic curve over $\mathbb{F}_q$ suitable for cryptographic purposes.

C Method Selecting the curve at random

1. Randomly select parameters $a, b \in \mathbb{F}_q$ to define the elliptic curve equation.
   In the case that q is a prime, verify that $4a^3 \quad 27b^2 \ / \ 0$. The curve equation is $E : y^2 \quad x^3 \quad ax \quad b$.
   In the case that $q \quad 2^m$, verify that $b \ / \ 0$. The curve equation is $E : y^2 \quad xy \quad x^3 \quad ax^2 \quad b$.
2. Compute $u \quad E(\mathbb{F}_q)$. (See notes below.)
3. Factor u if this order is not divisible by a large prime number r, then go to step 1.
4. Verify that the large prime divisor r of u does not divide $q^v - 1$, for $v \quad 1, 2, 3, \ldots, 10$. If this test fails, then go to step 1.
5. Output the curve selected.

Notes

The order $E(\mathbb{F}_q)$ can be computed by using Schoof s algorithm 23 . Although the basic algorithm is quite ine cient, several dramatic improvements and extensions of this method have been discovered during the last three years. Currently it is feasible to compute $E(\mathbb{F}_p)$ where p is as large as 10^{430} 13 . Also, it is possible to compute $E(\mathbb{F}_{2^m})$ where m is as big as 300 in a few hours on a workstation 18 .

C Method Selecting the order of the curve rst

1. Select an order u such that

   (a) $q \quad 1 - 2\sqrt{q} \leq u \leq q \quad 1 \quad 2\sqrt{q}$.
   (b) u is divisible by a large prime r.
   (c) r does not divide $q^v - 1$ for $v \quad 1, 2, 3, \ldots, 10$.

   If $q \quad 2^m$ then u should also be even.
2. Use the algorithm described in 12 to find parameters $a, b \in \mathbb{F}_q$ such that the elliptic curve E defined by them has order $E(\mathbb{F}_q) \quad u$.
3. Output the curve E.

Notes

The algorithm of 12 requires some precomputations for each particular field $\mathbb{F}_q$. Once this is done, then the algorithm takes a few minutes on a workstation, even when q is as large as 2^{300}.

C Method Using the Weil Theorem

This technique can be used for picking curves over $\mathbb{F}_{2^m}$, where m is divisible by a small number, say l.

1. Select a random curve $E : y^2 \quad xy \quad x^3 \quad ax^2 \quad b, b \ / \ 0$, where $a, b \in \mathbb{F}_{2^l}$. Note that since $\mathbb{F}_{2^l}$ is contained in $\mathbb{F}_{2^m}$, it is also true that $a, b \in \mathbb{F}_{2^m}$, and so E is also a curve over $\mathbb{F}_{2^m}$.
2. Compute $w \quad E(\mathbb{F}_{2^l})$. This can be done exhaustively since l is small.
3. Let $t \quad q^l \quad 1 - w$, and let $c \quad m/l$. Then

   $$u \quad E(\mathbb{F}_{2^m}) \quad 2^m \quad 1 - \alpha^c - \beta^c,$$

   where α and β are complex numbers determined from the factorization of

   $$1 - tz \quad q^l z^2 \quad (1 - \alpha z)(1 - \beta z).$$

4. Factor u if this order is not divisible by a large prime number r, then go to step 1.
5. Verify that the large prime divisor r of u does not divide $q^v - 1$, for $v \quad 1, 2, 3, \ldots, 10$. If this test fails, then go to step 1.
6. Output the curve selected.

C.8 Computing the Order of a Point

The following algorithm is an e cient method for computing the order of an elliptic curve point, given the prime factorization of the elliptic curve order.

Input: An elliptic curve E defined over $\mathbb{F}_q$, where the prime factorization of $E(\mathbb{F}_q)$ is

$$E(\mathbb{F}_q) \quad p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}, \quad e_i \geq 1.$$

A point P on E.
Output: The order n of P.

1. Set $n \longleftarrow \quad E(\mathbb{F}_q)$.
2. For i from 1 to k do

   2.1 Set $n \longleftarrow n/p_i^{e_i}$.
   2.2 Compute $P_1 \longleftarrow nP$.
   2.3 While $P_1 \ / \ \mathcal{O}$, compute $P_1 \longleftarrow p_i P_1$ and set $n \longleftarrow np_i$.

3. Output n.

C.9 Representing an Elliptic Curve Point

An elliptic curve point P (which is not the point at infinity $\mathcal{O}$) is represented by two field elements, the x-coordinate of P and the y-coordinate of P: $P \quad (x_P, y_P)$. The point can be represented more compactly by storing only the x-coordinate x_P and a certain bit $\widetilde{y_P}$. The next two subsections show how the full y-coordinate y_P can be recovered from x_P and $\widetilde{y_P}$.

C Elliptic curves over $\mathbb{F}_p$

We impose the condition that $p \equiv 3 \pmod 4$, that is $p = 4u + 3$ for some positive integer u. (The case where $p \equiv 1 \pmod 4$ is more complicated, and is not discussed here.)

Let $P = (x_P, y_P)$ be a point on the elliptic curve $E : y^2 = x^3 + ax + b$ defined over a prime field $\mathbb{F}_p$. Recall that $\widetilde{y_P}$ is equal to the least significant bit of y_P.

Suppose that we are given the x-coordinate x_P of P and the bit $\widetilde{y_P}$. Then y_P can be recovered as follows.

1. Compute the field element $\alpha = {x_P}^3 + ax_P + b \bmod p$.
2. Compute the field element $\beta = \alpha^{u+1} \bmod p$.
3. If the least significant bit of β is equal to $\widetilde{y_P}$ then set $y_P \longleftarrow \beta$. Otherwise, set $y_P \longleftarrow p - \beta$.

C Elliptic curves over $\mathbb{F}_{2^m}$

The technique described in this subsection works only if the elements of the field $\mathbb{F}_{2^m}$ are represented with respect to a normal basis representation (see Section C.6).

Let $P = (x_P, y_P)$ be a point on the elliptic curve $E : y^2 + xy = x^3 + ax^2 + b$ defined over a field $\mathbb{F}_{2^m}$. Recall that $\widetilde{y_P}$ is equal to 0 if $x_P = 0$ if $x_P \neq 0$ then $\widetilde{y_P}$ is equal to the least significant bit of the field element $y_P \cdot {x_P}^{-1}$.

Suppose that we are given the x-coordinate x_P of P and the bit $\widetilde{y_P}$. Then y_P can be recovered as follows.

1. If $x_P = 0$ then y_P is obtained by cyclically shifting the binary representation of the field element b one position to the left. That is, if $b = b_{m-1}b_{m-2}\ldots b_1 b_0$, then $y_P \longleftarrow b_{m-2}\ldots b_1 b_0 b_{m-1}$.
2. If $x_P \neq 0$ then do the following:

   2.1 Compute the field element $\alpha = x_P + a + bx_P^{-2}$ in $\mathbb{F}_{2^m}$.

   2.2 Let the binary representation of α be $\alpha = \alpha_{m-1}\alpha_{m-2}\ldots\alpha_1\alpha_0$.

   2.3 Construct a field element $z = z_{m-1}z_{m-2}\ldots z_1 z_0$ by setting

$$\begin{aligned} z_0 &= \widetilde{y_P}. \\ z_1 &= \alpha_0 \oplus z_0. \\ z_2 &= \alpha_1 \oplus z_1. \\ &\vdots \\ z_{m-2} &= \alpha_{m-3} \oplus z_{m-3}. \\ z_{m-1} &= \alpha_{m-2} \oplus z_{m-2}. \end{aligned}$$

   2.4 Finally, compute $y_P \longleftarrow x_P \cdot z$.

Appendix E

Validation Suite (Test Vectors)

Appendix F

Known State of Attacks

The security of the elliptic curve systems described in this standard hinges on the apparent di culty of the *discrete logarithm problem* in the elliptic curve. Namely, given an elliptic curve E defined over a finite field $\mathbb{F}_q$, and given points P and $Q(\quad kP)$, find the integer k.

Unlikely the case of the discrete logarithm problem in finite fields, there is no subexponential time algorithm known for the elliptic curve discrete logarithm problem (in the case that the curve is non-singular). The best algorithm known to date is the Pollard-ρ method 22 which takes about $\sqrt{\pi r/2}$ steps, where r is largest prime divisor of the order n of P.

Recently, van Oorschot and Wiener 21 discovered a technique for parallelizing the Pollard-ρ method so that if m processors are used, then the expected number of steps by each processor before a discrete logarithm is obtained is $\sqrt{\pi r/2}/m$. Hence, to avoid this attack, it is necessary to select a curve and a point P of order divisible by a prime r so that $\sqrt{\pi r/2}/m$ is su ciently large.

As a concrete example, van Oorschot and Wiener estimated that if r is about 10^{36}, then a machine with 325,000 processors can be built for about 10 million which would compute logarithms in about 35 days.

If r is chosen to be at least 10^{44}, then the discrete logarithm problem would be well out of reach of the van Oorschot and Wiener attack.

The special classes of supersingular curves have been avoided in this standard since there is a method for reducing the discrete logarithm problem in these curves to the discrete logarithm problem in a finite field. However, it should be pointed out that there are some particular supersingular curves whose use may have some advantages over non-supersingular curves.

References

Elliptic curve cryptosystems were first proposed in 1985 independently by Neil Koblitz 11 and Victor Miller 17 . Since then a lot of research has been done towards improving the e ciency of these systems, and evaluating their security. For a summary of this work, consult 16 . A description of a hardware implementation of an elliptic curve cryptosystem can be found in 2 .

Two good references on the theory of finite fields are the books by McEliece 15 and Lidl and Niederreiter 14 . The article 1 explains how to e ciently perform arithmetic operations in finite fields of characteristic 2. A hardware implementation of arithmetic in such fields which exploits the properties of so-called optimal normal bases is described in 3 .

The ElGamal public-key encryption and signature schemes were introduced in 6 . The NIST Digital Signature Standard (DSS) is described in 20 . The Secure Hash Algorithm (SHA) is described in 4 and 19 . Abstract Syntax Notation One (ASN.1), Basic Encoding Rules (BER) and Distinguished Encoding Rules (DER) are described in 8 , 9 and 10 respectively.

[1] G. Agnew, T. Beth, R. Mullin and S. Vanstone, "Arithmetic operations in $GF(2^m)$", *Journal of Cryptology*, **6** (1993), 3-13.

[2] G. Agnew, R. Mullin and S. Vanstone, "An implementation of elliptic curve cryptosystems over $F_{2^{155}}$", *IEEE Journal on Selected Areas in Communications*, **11** (1993), 804-813.

[3] G. Agnew, R. Mullin, I. Onyszchuk and S. Vanstone, "An implementation for a fast public-key cryptosystem", *Journal of Cryptology*, **3** (1991), 63-79.

[4] American National Standards Institute, "Public key cryptography using irreversible algorithms for the financial services industry: part 2: the secure hash algorithm (SHA)", ANSI X9.30-1993.

[5] E. Brickell, D. Gordon, K. McCurley and D. Wilson, "Fast exponentiation with precomputation", *Advances in Cryptology EUROCRYPT* , Lecture Notes in Computer Science, **658** (1993), Springer Verlag, 200-207.

[6] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, **31** (1985), 469-472.

[7] ISO/IEC 9796 "Information Technology Security Techniques Digital signature schemes giving message recovery".

[8] ISO/IEC 8824, "Information Technology Open Systems Interconnection Abstract syntax notation one (ASN.1)".

[9] ISO/IEC 8825-1, "Information Technology Open Systems Interconnection Specification of ASN.1 encoding rules Part 1: Basic Encoding Rules (BER)".

[10] ISO/IEC 8825-3, "Information Technology Open Systems Interconnection Specification of ASN.1 encoding rules Part 3: Distinguished canonical encoding rules".

[11] N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, **48** (1987), 203-209.

[12] G. Lay and H. Zimmer, "Constructing elliptic curves with given group order over large finite fields", presented at the First Algorithmic Number Theory Symposium, Ithaca, May 1994.

[13] F. Lehmann, M. Maurer, V. Muller and V. Shoup, "Counting the number of points on elliptic curves over finite fields of characteristic greater than three", presented at the First Algorithmic Number Theory Symposium, Ithaca, May 1994.

[14] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press, 1987.

[15] R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, 1987.

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

[17] V. Miller, "Uses of elliptic curves in cryptography", *Advances in Cryptology CRYPTO* , Lecture Notes in Computer Science, **218** (1986), Springer-Verlag, 417-426.

[18] F. Morain, private communication, October 4, 1994.

[19] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS Publication 180, May 1993.

[20] National Institute of Standards and Technology, "Digital signature standard", FIPS Publication 186, 1993.

[21] P. van Oorschot and M. Wiener, "Parallel collision search with application to hash functions and discrete logarithms", to be presented at the 2nd ACM Conference on Computer and Communications Security, Fairfax, Virginia, November 4, 1994.

[22] J. Pollard, "Monte Carlo methods for index computation mod p", *Mathematics of Computation*, **32** (1978), 918-924.

[23] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod p", *Mathematics of Computation*, **44** (1985), 483-494.