

# PSEC-KEM Specification

NTT Information Sharing Platform Laboratories, NTT Corporation

September 26, 2001

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Overview . . . . .	3
<b>2</b>	<b>Notation</b>	<b>3</b>
<b>3</b>	<b>Data types and conversions</b>	<b>4</b>
3.1	BitString-to-OctetString Conversion(BS2OSP) . . . . .	4
3.2	OctetString-to-BitString Conversion(OS2BSP) . . . . .	5
3.3	Integer-to-OctetString Conversion(I2OSP) . . . . .	5
3.4	OctetString-to-Integer Conversion(OS2IP) . . . . .	6
3.5	Field Element-to-Integer Conversion(FE2IP) . . . . .	6
3.6	Integer-to-Field Element Conversion(I2FEP) . . . . .	7
3.7	FieldElement-to-OctetString Conversion(FE2OSP) . . . . .	7
3.8	OctetString-to-FieldElement Conversion(OS2FEP) . . . . .	8
3.9	EllipticCurvePoint-to-OctetString Conversion (ECP2OSP) . . . . .	8
3.10	OctetString-to-EllipticCurvePoint Conversion(OS2ECPP) . . . . .	9
<b>4</b>	<b>Key types</b>	<b>10</b>
4.1	PSEC private key . . . . .	10
4.2	PSEC public key . . . . .	10
<b>5</b>	<b>Cryptographic primitives</b>	<b>11</b>
5.1	KGP-PSEC . . . . .	11
5.2	EP-PSEC . . . . .	12
5.3	DP-PSEC . . . . .	12
<b>6</b>	<b>Key encapsulation mechanisms</b>	<b>12</b>
6.1	ES-PSEC-KEM . . . . .	13
6.1.1	Encryption operation . . . . .	13
6.1.2	Decryption operation . . . . .	13
<b>7</b>	<b>Encoding methods</b>	<b>13</b>
7.1	EME-PSEC-KEM . . . . .	14
7.1.1	Encoding operation EME-PSEC-KEM-A . . . . .	14
7.1.2	Encoding operation EME-PSEC-KEM-B . . . . .	14
7.1.3	Decoding operation EME-PSEC-KEM-C . . . . .	14
7.1.4	Decoding operation EME-PSEC-KEM-D . . . . .	15

<b>8</b>	<b>Auxiliary techniques</b>	<b>15</b>
8.1	Hash functions . . . . .	15
8.1.1	SHA-1 . . . . .	15
8.2	Key derivation functions . . . . .	15
8.2.1	MGF1 . . . . .	16
<b>A</b>	<b>Design principles</b>	<b>17</b>
<b>B</b>	<b>Security requirements of parameters</b>	<b>17</b>
<b>C</b>	<b>Recommendation values of parameters</b>	<b>17</b>
<b>D</b>	<b>Fast implementation techniques</b>	<b>17</b>
<b>E</b>	<b>Useful information for implementations</b>	<b>18</b>
<b>F</b>	<b>Information about versions</b>	<b>18</b>
F.1	Reasons for changing recommended parameters . . . . .	18
F.2	User's disadvantage without compatibility . . . . .	18

## 1 Introduction

This document provides recommendations for implementation of the PSEC-KEM key encapsulation mechanism. We can utilize the mechanism for key agreement schemes. This document covers the following issues:

- cryptographic primitives: KGP-PSEC, EP-PSEC, DP-PSEC
- key encapsulation mechanisms: ES-PSEC-KEM

### 1.1 Overview

The organization of this document is as follows:

- Section 1 is an introduction.
- Section 2 defines some notations.
- Section 3 defines some data types and conversions.
- Section 4 defines the PSEC private and public keys, which are used in KGP-PSEC, EP-PSEC, DP-PSEC.
- Section 5 defines several cryptographic primitives, for the PSEC-KEM key encapsulation mechanism.
- Section 6 defines the key encapsulation mechanism.
- Section 7 defines the encoding method for the key encapsulation mechanism.
- Section 8 defines some auxiliary functions used in this document.

## 2 Notation

$N$	the set of natural integers
$\mathbb{F}_{q^m}$	a finite field with $q^m$ elements, where $q$ is a prime.
$\mathcal{O}$	a point at infinity on an elliptic curve
$\{0, 1\}^i$	the set of all bit strings of length $i$
$\{0, 1\}^*$	$\bigcup_{i=0}^{\infty} \{0, 1\}^i$
$\{0, 1, \dots, 255\}^i$	the set of all octet strings of length $i$
$\{0, 1, \dots, 255\}^*$	$\bigcup_{i=0}^{\infty} \{0, 1, \dots, 255\}^i$
$\parallel$	a concatenation operator for two bit strings or a concatenation operator for two octet strings, for example, $(0, 1, 0, 0) \parallel (1, 1, 0) = (0, 1, 0, 0, 1, 1, 0)$ for bit strings, $(4, 3) \parallel (6, 2) = (4, 3, 6, 2)$ for octet strings
$\oplus$	the bit-wise exclusive-or operation
$\lceil y \rceil$	the least integer greater than or equal to $y$
$\lfloor y \rfloor$	the greatest integer less than or equal to $y$
$a \bmod m$	the least non negative integer $b$ which satisfies $m \mid (b - a)$ for $a, m \in N$

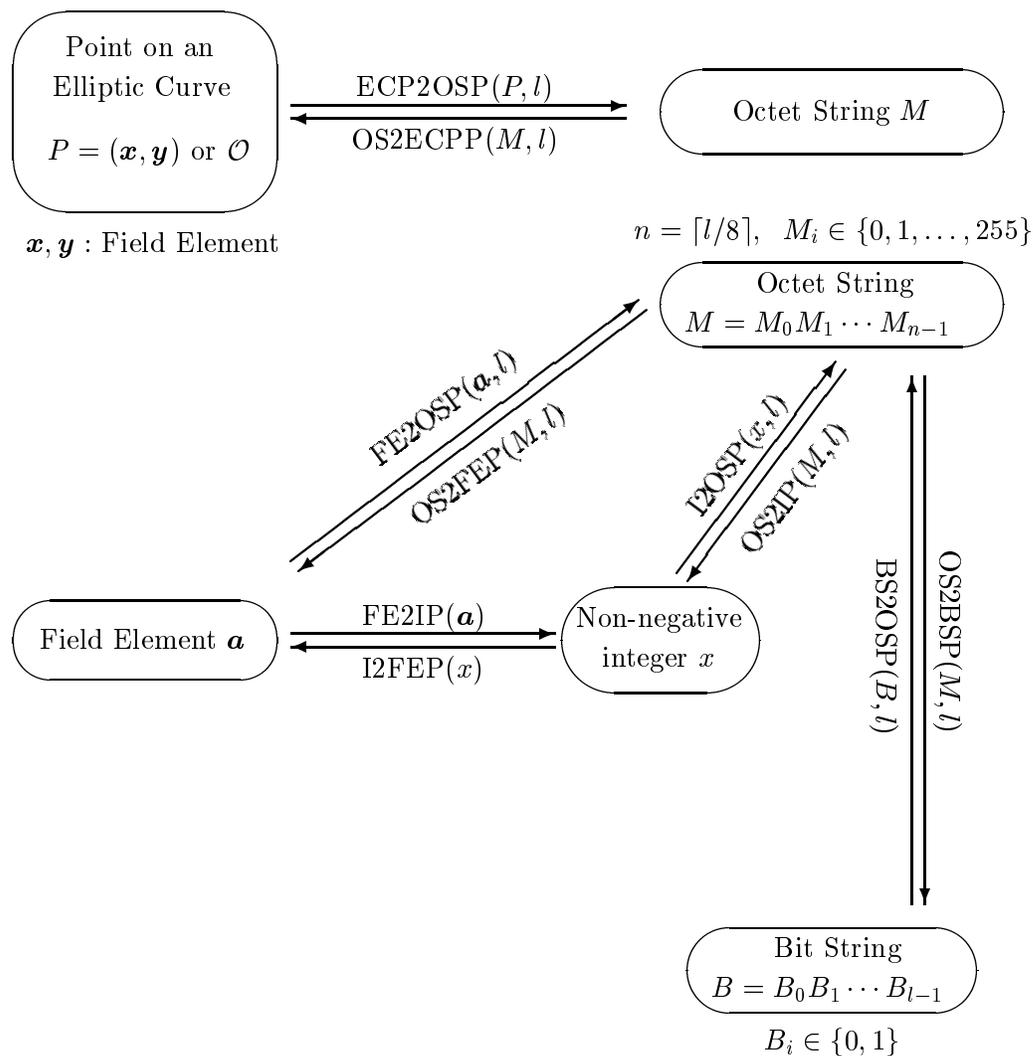


Figure 1: Converting between data types

The concatenation operator '||' is often omitted.

### 3 Data types and conversions

The schemes specified in this document involve operations using several different data types. Figure 1 illustrates which conversions are needed and where they are described.

#### 3.1 BitString-to-OctetString Conversion(BS2OSP)

Bit strings should be converted to octet strings as described in this section. Informally the idea is to pad the bit string with 0's on the left to make its length a multiple of 8, then chop the result up into octets. Formally the conversion routine,  $\text{BS2OSP}(B, l)$ , is specified as follows:

**Input:**

$B$  : a bit string of length  $l$  bits  
 $l$  : an integer

**Output:**

$M$  : an octet string of length  $n = \lceil l/8 \rceil$  octets

**Steps:**

Convert the bit string  $B = B_0B_1 \dots B_{l-1}$  to an octet string  $M = M_0M_1 \dots M_{n-1}$  as follows:

1. For  $0 < i \leq n - 1$ , let:

$$M_i := B_{l-8-8(n-1-i)}B_{l-7-8(n-1-i)} \dots B_{l-1-8(n-1-i)}$$

2. Set the leftmost  $8n - l$  bits in  $M_0$  to 0's, and the rightmost  $l + 8 - 8n$  bits to  $B_0B_1 \dots B_{l+7-8n}$ .
3. Output  $M$ .

**3.2 OctetString-to-BitString Conversion(OS2BSP)**

Octet strings should be converted to bit strings as described in this section. Informally the idea is simply to view the octet string as a bit string instead. Formally the conversion routine,  $\text{OS2BSP}(M, l)$ , is specified as follows:

**Input:**

$M$  : an octet string of length  $n = \lceil l/8 \rceil$  octets  
 $l$  : an integer

**Output:**

$B$  : a bit string of length  $l$  bits

**Steps:**

Convert the octet string  $M = M_0M_1 \dots M_{n-1}$  to a bit string  $B = B_0B_1 \dots B_{l-1}$  as follows:

1. For  $0 < i \leq n - 1$ , set:

$$B_{l-8-8(n-1-i)}B_{l-7-8(n-1-i)} \dots B_{l-1-8(n-1-i)} := M_i.$$

2. Ignore the leftmost  $8n - l$  bits of  $M_0$ , and set  $B_0B_1 \dots B_{l+7-8n}$  to the rightmost  $l + 8 - 8n$  bits of  $M_0$ .
3. Output  $B$ .

**3.3 Integer-to-OctetString Conversion(I2OSP)**

Integers should be converted to octet strings as described in this section. Informally the idea is to represent the integer in binary then convert the resulting bit string to an octet string. Formally the conversion routine,  $\text{I2OSP}(x, l)$ , is specified as follows:

**Input:**

$x$  : a nonnegative integer  
 $l$  : an integer

**Output:**

$M$  : an octet string of length  $n = \lceil l/8 \rceil$  octets

**Errors:**

“invalid”

**Steps:**

1. If  $x \geq 2^l$ , assert “invalid” and stop.
2. Determine the  $x$ 's base-256 representation,  $x_i \in \{0, \dots, 255\}$  such that

$$x = x_{n-1}2^{8(n-1)} + x_{n-2}2^{8(n-2)} + \dots + x_12^8 + x_0$$

3. For  $0 \leq i \leq n-1$ , set  $M_i = x_{n-1-i}$ , and let

$$M := M_0M_1 \dots M_{n-1}.$$

4. Output  $M$ .

**3.4 OctetString-to-Integer Conversion(OS2IP)**

Octet strings should be converted to integers as described in this section. Informally the idea is simply to view the octet string as the base 256 representation of the integer. Formally the conversion routine,  $\text{OS2IP}(M, l)$ , is specified as follows:

**Input:**

$M$  : an octet string of length  $n = \lceil l/8 \rceil$  octets  
 $l$  : an integer

**Output:**

$x$  : an integer

**Steps:**

Convert  $M = M_0M_1 \dots M_{n-1}$  to an integer  $x$  as follows:

1. View each  $M_i$  as an integer in  $\{0, \dots, 255\}$ , and output  $x$  such that

$$x := \sum_{i=0}^{n-1} 2^{8(n-1-i)} M_i \pmod{2^l}.$$

**3.5 Field Element-to-Integer Conversion(FE2IP)**

Field elements should be converted to integers as described in this section. A field element should be represented as a polynomial with integer coefficients, which can be represented as a sequence of the coefficients. Informally the idea is simply to view the sequence of the coefficients as the radix- $q$  representation of the integer, where  $q$  is the characteristic of the field. Formally the conversion routine,  $\text{FE2IP}(\mathbf{a})$ , is specified as follows:

**System Parameters:**

$\mathbb{F}_{q^m}$  : a finite field with  $q^m$  elements where  $q$  is a prime, and  $m > 0$  is an integer

**Input:**

$\mathbf{a}$  : a field element in  $\mathbb{F}_{q^m}$

**Output:**

$x$  : an integer in  $\{0, \dots, q^m - 1\}$

**Steps:**

Convert the field element  $\mathbf{a}$  to a integer  $x$  as follows:

if  $m = 1$ :

The field element  $\mathbf{a}$  must be represented as an integer in  $\{0, \dots, q^m - 1\}$ .

1: Output  $x$  such that  $x := \mathbf{a}$

if  $m > 1$ :

The field element  $\mathbf{a}$  must be represented as a polynomial of at most  $(m-1)$ -th degree with coefficients in  $\{0, \dots, q-1\}$ . Let  $\beta$  be the variable of the polynomial.

1: Determine the coefficients  $a_i \in \{0, \dots, q-1\}$  for  $i \in \{0, \dots, m-1\}$  which satisfy

$$\mathbf{a} = \sum_{i=0}^{m-1} a_i \beta^i$$

2: Output  $x$  such that

$$x := \sum_{i=0}^{m-1} a_i q^i$$

### 3.6 Integer-to-Field Element Conversion(I2FEP)

Integers should be converted to field elements as described in this section. A field element should be represented as a polynomial with integer coefficients, and it can be represented as a sequence of the coefficients. Informally the idea is to represent the integer with radix- $q$  positional number system where  $q$  is the characteristic of the field, and then convert the each digit to the each coefficient of the polynomial. Formally the conversion routine,  $\text{I2FEP}(x)$ , is specified as follows:

#### System Parameters:

$\mathbb{F}_{q^m}$  : a finite field with  $q^m$  elements where  $q$  is a prime, and  $m > 0$  is an integer

#### Input:

$x$  : an integer in  $\{0, \dots, q^m - 1\}$

#### Output:

$\mathbf{a}$  : a field element in  $\mathbb{F}_{q^m}$

#### Steps:

Convert the integer  $x$  to a field element  $\mathbf{a}$  as follows:

if  $m = 1$ :

A field element of  $\mathbb{F}_{q^m}$  must be represented as an integer in  $\{0, \dots, q^m - 1\}$ .

1: Output  $\mathbf{a}$  such that  $\mathbf{a} := x$ .

if  $m > 1$ :

A field element of  $\mathbb{F}_{q^m}$  must be represented as a polynomial of at most  $(m-1)$ -th degree with coefficients in  $\{0, \dots, q-1\}$ . Let  $\beta$  be the variable of the polynomial.

1: Expand  $x$  into it's radix  $q$  representation  $x_i \in \{0, \dots, q-1\}$  for  $i \in \{0, \dots, m-1\}$  which satisfy

$$x = \sum_{i=0}^{m-1} x_i q^i$$

2: Output  $\mathbf{a}$  such that

$$\mathbf{a} := \sum_{i=0}^{m-1} x_i \beta^i$$

### 3.7 FieldElement-to-OctetString Conversion(FE2OSP)

The conversion routine,  $\text{FE2OSP}(\mathbf{a}, l)$ , is specified as follows:

**Input:**

$\mathbf{a}$  : a field element  
 $l$  : an integer

**Output:**

$M$  : an octet string

**Steps:**

Output  $M$  such that

$$M := \text{I2OSP}(\text{FE2IP}(\mathbf{a}), l).$$

**3.8 OctetString-to-FieldElement Conversion(OS2FEP)**

The conversion routine,  $\text{OS2FEP}(M, l)$ , is specified as follows:

**Input:**

$M$  : an octet string  
 $l$  : an integer

**Output:**

$\mathbf{a}$  : a field element

**Steps:**

Output  $\mathbf{a}$  such that

$$\mathbf{a} := \text{I2FEP}(\text{OS2IP}(M, l)).$$

**3.9 EllipticCurvePoint-to-OctetString Conversion (ECP2OSP)**

Elliptic curve points should be converted to octet strings as described in this section. Informally, if point compression is being used, the idea is that the compressed  $y$ -coordinate is placed in the leftmost octet of the octet string along with an indication that point compression is on, and the  $x$ -coordinate is placed in the remainder of the octet string; otherwise if point compression is off, the leftmost octet indicates that point compression is off, and remainder of the octet string contains the  $x$ -coordinate followed by the  $y$ -coordinate. Formally the conversion routine,  $\text{ECP2OSP}(P, l)$ , is specified as follows:

**Options:**

$E$  : an elliptic curve parameter  
 $R$  : Compressed, Uncompressed, or Hybrid

**Input:**

$P$  : a point on an elliptic curve over  $\mathbb{F}_{q^m}$   
 $l$  : an integer

**Output:**

$M$  : an octet string of length  $n$

$$\text{where } \begin{cases} n = 1 & \text{if } P = \mathcal{O}, \\ n = \lceil l/8 \rceil + 1 & \text{if } P \neq \mathcal{O} \text{ and } R \text{ is Compressed,} \\ n = 2\lceil l/8 \rceil + 1 & \text{if } P \neq \mathcal{O} \text{ and } R \text{ is Uncompressed or Hybrid.} \end{cases}$$

**Steps:**

Convert  $P$  to an octet string  $M = M_0M_1 \dots M_{n-1}$  as follows:

1. If  $P = \mathcal{O}$ , output  $M := 00_{16}$ .
2. If  $P = (\mathbf{x}, \mathbf{y}) \neq \mathcal{O}$  and  $R = \text{Compressed}$ , proceed as follows:
  - 2.1. Set an octet string  $X := \text{FE2OSP}(\mathbf{x}, l)$ .

- 2.2. Derive from  $\mathbf{y}$  a single bit  $\tilde{y}$  as follows (this allows the  $y$ -coordinate to be represented compactly using a single bit):
  - 2.2.1. If  $q$  is an odd number, set  $\tilde{y} := y_0 \bmod 2$  where  $\mathbf{y} = y_{m-1}\beta^{m-1} + \dots + y_1\beta + y_0$ .
  - 2.2.2. If  $q = 2$ , set  $\tilde{y} := 0$  if  $\mathbf{x} = \mathbf{0}$ , otherwise compute  $\mathbf{z} = z_{m-1}\beta^{m-1} + \dots + z_1\beta + z_0$  such that  $\mathbf{z} = \mathbf{y}\mathbf{x}^{-1}$  and set  $\tilde{y} := z_0$ .
- 2.3. If  $\tilde{y} = 0$ , assign the value  $02_{16}$  to the single octet  $L$ . If  $\tilde{y} = 1$ , assign the value  $03_{16}$  to the single octet  $L$ .
- 2.4. Output  $M := L\|X$ .
3. If  $P = (\mathbf{x}, \mathbf{y}) \neq \mathcal{O}$  and  $R = \text{Uncompressed}$ , proceed as follows:
  - 3.1. Set an octet string  $X := \text{FE2OSP}(\mathbf{x}, l)$ .
  - 3.2. Set an octet string  $Y := \text{FE2OSP}(\mathbf{y}, l)$ .
  - 3.3. Output  $M := 04_{16}\|X\|Y$ .
4. If  $P = (\mathbf{x}, \mathbf{y}) \neq \mathcal{O}$  and  $R = \text{Hybrid}$ , proceed as follows:
  - 4.1. Set an octet string  $X := \text{FE2OSP}(\mathbf{x}, l)$ .
  - 4.2. Set an octet string  $Y := \text{FE2OSP}(\mathbf{y}, l)$ .
  - 4.3. Derive from  $\mathbf{y}$  a single bit  $\tilde{y}$  as follows (this allows the  $y$ -coordinate to be represented compactly using a single bit):
    - 4.3.1. If  $q$  is an odd number, set  $\tilde{y} := y_0 \bmod 2$  where  $\mathbf{y} = y_{m-1}\beta^{m-1} + \dots + y_1\beta + y_0$ .
    - 4.3.2. If  $q = 2$ , set  $\tilde{y} := 0$  if  $\mathbf{x} = \mathbf{0}$ , otherwise compute  $\mathbf{z} = z_{m-1}\beta^{m-1} + \dots + z_1\beta + z_0$  such that  $\mathbf{z} = \mathbf{y}\mathbf{x}^{-1}$  and set  $\tilde{y} := z_0$ .
  - 4.4. If  $\tilde{y} = 0$ , assign the value  $06_{16}$  to the single octet  $L$ . If  $\tilde{y} = 1$ , assign the value  $07_{16}$  to the single octet  $L$ .
  - 4.5. Output  $M := L\|X\|Y$ .

### 3.10 OctetString-to-EllipticCurvePoint Conversion(OS2ECP)

Octet strings should be converted to elliptic curve points as described in this section. Informally the idea is that, if the octet string represents a compressed point, the compressed  $y$ -coordinate is recovered from the leftmost octet, the  $x$ -coordinate is recovered from the remainder of the octet string, and then the point compression process is reversed; otherwise the leftmost octet of the octet string is removed, the  $x$ -coordinate is recovered from the left half of the remaining octet string, and the  $y$ -coordinate is recovered from the right half of the remaining octet string. Formally the conversion routine,  $\text{OS2ECP}(M, l)$ , is specified as follows:

**Option:**

$E$  : an elliptic curve parameter

**Input:**

$M$  : an octet string which is either  
     the single octet  $00_{16}$ ,  
     an octet string of length  $n = \lceil l/8 \rceil + 1$ , or  
     an octet string of length  $n = 2\lceil l/8 \rceil + 1$

$l$  : an integer

**Output:**

$P$  : an elliptic curve point

**Errors:**

“invalid”

**Steps:**

Convert  $M$  to an elliptic curve point  $P$  as follows:

1. If  $M = 00_{16}$ , output  $P := \mathcal{O}$ .
2. If  $M$  has length  $\lceil l/8 \rceil + 1$  octets, proceed as follows:
  - 2.1. Parse  $M = L\|X$  as a single octet  $L$  followed by  $\lceil l/8 \rceil$  octets  $X$ .
  - 2.2. Set  $\mathbf{x} := \text{OS2FEP}(X, l)$ .
  - 2.3. If  $L = 02_{16}$ , set  $\tilde{y} := 0$ , and if  $L = 03_{16}$ , set  $\tilde{y} := 1$ . Otherwise assert “invalid” and stop.
  - 2.4. Derive from  $\mathbf{x}$  and  $\tilde{y}$  an elliptic curve point  $P := (\mathbf{x}, \mathbf{y})$ , where:
    - 2.4.1. If  $q$  is an odd number, compute the field element  $\mathbf{w} := \mathbf{x}^3 + \mathbf{a}\mathbf{x} + \mathbf{b}$ , and compute a square root  $\boldsymbol{\gamma}$  of  $\mathbf{w}$  in  $\mathbb{F}_{q^m}$ . Assert “invalid” and stop if there are no square roots in  $\mathbb{F}_{q^m}$ , otherwise set  $\mathbf{y} := \boldsymbol{\gamma}$  if  $\beta_0 \equiv \tilde{y} \pmod{2}$ , and set  $\mathbf{y} := -\boldsymbol{\gamma}$  if  $\beta_0 \not\equiv \tilde{y} \pmod{2}$ , where  $\boldsymbol{\gamma} = \gamma_{m-1}\beta^{m-1} + \dots + \gamma_1\beta + \gamma_0$ .
    - 2.4.2. If  $q = 2$  and  $\mathbf{x} = \mathbf{0}$ , set  $\mathbf{y} := \mathbf{b}^{2^{m-1}}$  in  $\mathbb{F}_{q^m}$ .
    - 2.4.3. If  $q = 2$  and  $\mathbf{x} \neq \mathbf{0}$ , compute the field element  $\boldsymbol{\gamma} := \mathbf{x} + \mathbf{a} + \mathbf{b}\mathbf{x}^{-2}$  in  $\mathbb{F}_{q^m}$ , and find an element  $\mathbf{z} = z_{m-1}\beta^{m-1} + \dots + z_1\beta + z_0$  such that  $\mathbf{z}^2 + \mathbf{z} = \boldsymbol{\gamma}$  in  $\mathbb{F}_{q^m}$ . Assert “invalid” and stop if no such  $\mathbf{z}$  exists, otherwise set  $\mathbf{y} := \mathbf{x}\mathbf{z}$  in  $\mathbb{F}_{q^m}$  if  $z_0 = \tilde{y}$ , and set  $\mathbf{y} := \mathbf{x}(\mathbf{z} + \mathbf{1})$  in  $\mathbb{F}_{q^m}$  if  $z_0 \neq \tilde{y}$ .
  - 2.5. Output  $P := (\mathbf{x}, \mathbf{y})$ .
3. If  $M$  has length  $2\lceil l/8 \rceil + 1$  octets, proceed as follows:
  - 3.1. Parse  $M = L\|X\|Y$  as a single octet  $L$  followed by  $\lceil l/8 \rceil$  octets  $X$  followed by  $\lceil l/8 \rceil$  octets  $Y$ .
  - 3.2. Check that  $L = 04_{16}$  or  $06_{16}$  or  $07_{16}$ . If  $L \neq 04_{16}$  or  $06_{16}$  or  $07_{16}$ , assert “invalid” and stop.
  - 3.3. set  $\mathbf{x} := \text{OS2FEP}(X, l)$ .
  - 3.4. set  $\mathbf{y} := \text{OS2FEP}(Y, l)$ .
  - 3.5. If  $P := (\mathbf{x}, \mathbf{y})$  does not satisfy the defining equation of the elliptic curve  $E$ , then assert “invalid” and stop.
  - 3.6. Output  $P := (\mathbf{x}, \mathbf{y})$ .

## 4 Key types

In this section two types of keys are defined; PSEC private key and PSEC public key, which are used in three cryptographic primitives (KGP-PSEC, EP-PSEC, DP-PSEC) of PSEC-KEM.

### 4.1 PSEC private key

A PSEC private key is a following value:

- $s$ , a nonnegative integer

### 4.2 PSEC public key

A PSEC public key is the 4-tuple  $(E, W, KDF, hLen)$ , where the components have the following meanings:

- $E$ , an elliptic curve parameter

- $W$ , a point on  $E$
- $KDF$ , the choice from key derivation functions
- $hLen$ , a nonnegative integer

An elliptic curve parameter  $E$  is the 9-tuple  $(q, m, f(\beta), \mathbf{a}, \mathbf{b}, P, p, pLen, qmLen)$ , where the components have the following meanings:

- $q$ , a prime number
- $m$ , a positive integer
- $f(\beta)$ , a monic irreducible polynomial of degree  $m$  over  $\mathbb{F}_{q^m}$
- $\mathbf{a}$ , an element in  $\mathbb{F}_{q^m}$
- $\mathbf{b}$ , an element in  $\mathbb{F}_{q^m}$
- $P$ , a point on an elliptic curve

–  $\mathbf{x}$ , an element in  $\mathbb{F}_{q^m}$

–  $\mathbf{y}$ , an element in  $\mathbb{F}_{q^m}$

$$\mathbf{y}^2 = \mathbf{x}^3 + \mathbf{a}\mathbf{x} + \mathbf{b} \quad (q > 3)$$

$$\mathbf{y}^2 + \mathbf{x}\mathbf{y} = \mathbf{x}^3 + \mathbf{a}\mathbf{x}^2 + \mathbf{b} \quad (q = 2)$$

- $p$ , a prime, the order of  $P$
- $pLen$ , the value of  $\lceil \log_2 p \rceil$
- $qmLen$ , the value of  $\lceil \log_2 q^m \rceil$

In a valid PSEC public key, it holds that  $W = sP$ , where  $s$  is a PSEC private key in section 4.1.

**Note :**

A key derivation function  $KDF$  shall be one of the key derivation functions in Section 8.2.

## 5 Cryptographic primitives

In this section, three cryptographic primitives are specified.

### 5.1 KGP-PSEC

KGP-PSEC( $E, KDF, hLen$ ) is defined as follows:

- Input :**
- $E$  an elliptic curve parameter
  - $KDF$  the choice from key derivation functions
  - $hLen$  a nonnegative integer
- Output :**
- $PK$  PSEC public key,  $(E, W, KDF, hLen)$
  - $s$  PSEC private key, a nonnegative integer,  $0 \leq s < p$

**Steps :**

1. Generate a random integer  $s \in \{0, \dots, p-1\}$ .
2. Let  $W = sP$ .
3. Output  $PK = (E, W, KDF, hLen)$  and  $s$ .

## 5.2 EP-PSEC

EP-PSEC( $PK, \alpha$ ) is defined as follows:

- Input :**             $PK$    PSEC public key  
                        $\alpha$     random value, a nonnegative integer,  $0 \leq \alpha < p$
- Output :**            $Q$      a point on  $E$   
                        $C_1$     a point on  $E$
- Assumptions :**   public key  $PK$  is valid.
- Steps :**
1. Let  $Q = \alpha W$ .
  2. Let  $C_1 = \alpha P$ .
  3. Output  $(Q, C_1)$ .

## 5.3 DP-PSEC

DP-PSEC( $PK, C_1, s$ ) is defined as follows:

- Input :**             $PK$    PSEC public key  
                        $C_1$     a point on  $E$   
                        $s$      PSEC private key, a nonnegative integer,  $0 \leq s < p$
- Output :**            $Q$      a point on  $E$
- Assumptions :**   public key  $PK$  and private key  $s$  are valid.
- Steps :**
1. Let  $Q = sC_1$ .
  2. Output  $Q$ .

# 6 Key encapsulation mechanisms

A key encapsulation mechanism works just like a public-key encryption scheme, except that the encryption algorithm takes no input other than the recipient's public key. Instead, the encryption algorithm generates a pair  $(k, c_0)$ , where  $k$  is a octet string of some specified length, and  $c_0$  is an encryption of  $k$ , that is, the decryption algorithm applied to  $c_0$  yields  $k$ .

One can always use a public-key encryption scheme for this purpose, generating a random octet string, and then encrypting it under the recipient's public key. However, as we shall see, one can construct a key encapsulation scheme in other, more efficient, ways as well.

A key encapsulation mechanism PSEC-KEM consists of two operations.

- An encryption operation ES-PSEC-KEM-ENCRYPT( $PK$ ) that takes as input a public key  $PK$  and outputs a key/ciphertext pair  $(k, c_0)$ .

- A decryption operation  $\text{ES-PSEC-KEM-DECRYPT}(PK, s, c_0)$  that takes as input a public key  $PK$ , private key  $s$  and a ciphertext  $c_0$ , and outputs a key  $k$ .

## 6.1 ES-PSEC-KEM

### 6.1.1 Encryption operation

$\text{ES-PSEC-KEM-ENCRYPT}(PK)$  is defined as follows:

**Input :**  $PK$  PSEC public key

**Output :**  $c_0$  an octet string

$k$  an octet string

**Assumptions :** public key  $PK$  is valid.

**Steps :**

1. Let  $(\alpha, k, r) = \text{EME-PSEC-KEM-A}(PK)$  (see Section 7.1.1).
2. Let  $(Q, C_1) = \text{EP-PSEC}(PK, \alpha)$  (see Section 5.2).
3. Let  $c_0 = \text{EME-PSEC-KEM-B}(PK, Q, C_1, r)$  (see Section 7.1.2).
4. Output  $(c_0, k)$ .

### 6.1.2 Decryption operation

$\text{ES-PSEC-KEM-DECRYPT}(PK, s, c_0)$  is defined as follows:

**Input :**  $PK$  PSEC public key

$s$  PSEC private key, a nonnegative integer,  $0 \leq s < p$

$c_0$  an octet string

**Output :**  $k'$  an octet string

**Errors :** “invalid”

**Assumptions :** public key  $PK$  and private key  $s$  are valid.

**Steps :**

1. Let  $(C_1, c_2, g) = \text{EME-PSEC-KEM-C}(PK, c_0)$  (see Section 7.1.3).  
If the decoding operation returns “invalid,” then assert “invalid” and stop.
2. Let  $Q' = \text{DP-PSEC}(PK, C_1, s)$  (see Section 5.3).
3. Let  $(\alpha', k') = \text{EME-PSEC-KEM-D}(PK, c_2, g, Q')$  (see Section 7.1.4).
4. Check  $C_1 = \text{DP-PSEC}(PK, P, \alpha')$  (see Section 5.3). If it holds, output  $k'$ . Otherwise, assert “invalid” and stop.

## 7 Encoding methods

This section specifies one encoding method for the key encapsulation mechanism.

## 7.1 EME-PSEC-KEM

### 7.1.1 Encoding operation EME-PSEC-KEM-A

EME-PSEC-KEM-A( $PK$ ) is defined as follows:

**Option :**  $keyLen$  a nonnegative integer  
**Input :**  $PK$  PSEC public key  
**Output :**  $\alpha$  a nonnegative integer,  $0 \leq \alpha < p$   
 $k$  an octet string  
 $r$  an octet string

**Steps :**

1. Generate a random octet string  $r \in \{0, \dots, 255\}^{\lceil hLen/8 \rceil}$ .
2. Let  $H = \text{OS2BSP}(KDF(\text{I2OSP}(0, 32) \parallel r, pLen + 128 + keyLen), pLen + 128 + keyLen)$ .
3. Parse  $H = t \parallel k'$ , where the bit length of  $t$  is  $pLen + 128$ , the bit length of  $k'$  is  $keyLen$ .
4. Let  $\alpha = \text{BS2IP}(t, pLen + 128) \bmod p$ .
5. Let  $k = \text{BS2OSP}(k', keyLen)$ .
6. Output  $(\alpha, k, r)$ .

### 7.1.2 Encoding operation EME-PSEC-KEM-B

EME-PSEC-KEM-B( $PK, Q, C_1, r$ ) is defined as follows:

**Option:**  $R$  Compressed, Uncompressed or Hybrid  
**Input :**  $PK$  PSEC public key  
 $Q$  a point on  $E$   
 $C_1$  a point on  $E$   
 $r$  an octet string  
**Output :**  $c_0$  an octet string

**Steps :**

1. Let  $c_2 = r \oplus KDF(\text{I2OSP}(1, 32) \parallel \text{ECP2OSP}(C_1, qmLen) \parallel \text{ECP2OSP}(Q, qmLen), hLen)$ .
2. Let  $c_0 = \text{ECP2OSP}(C_1, qmLen) \parallel c_2$ .
3. Output  $c_0$ .

### 7.1.3 Decoding operation EME-PSEC-KEM-C

EME-PSEC-KEM-C( $PK, c_0$ ) is defined as follows:

**Option :**  $R$  Compressed, Uncompressed or Hybrid  
**Input :**  $PK$  PSEC public key  
 $c_0$  an octet string  
**Output :**  $C_1$  a point on  $E$   
 $c_2$  an octet string  
 $g$  an octet string  
**Errors :** “invalid”  
**Steps :**

1. If the octet length of  $c_0$  is less than or equal to  $\lceil hLen/8 \rceil$ , assert “invalid” and stop.
2. Parse  $c_0 = g || c_2$ , where the octet length of  $c_2$  is  $\lceil hLen/8 \rceil$ .
3. Let  $C_1 = \text{OS2ECP}(g, qmLen)$ .  
If OS2ECP asserts “invalid,” assert “invalid” and stop.
4. Output  $(C_1, c_2, g)$ .

#### 7.1.4 Decoding operation EME-PSEC-KEM-D

EME-PSEC-KEM-D( $PK, c_2, g, Q'$ ) is defined as follows:

**Option :**  $keyLen$  a nonnegative integer  
**Input :**  $PK$  PSEC public key  
 $c_2$  an octet string  
 $g$  an octet string  
 $Q'$  a point on  $E$   
**Output :**  $\alpha'$  a nonnegative integer,  $0 \leq \alpha' < p$   
 $k'$  an octet string

**Steps :**

1. Let  $r' = c_2 \oplus \text{KDF}(\text{I2OSP}(1, 32) || g || \text{ECP2OSP}(Q', qmLen), hLen)$ .
2. Let  $h' = \text{OS2BSP}(\text{KDF}(\text{I2OSP}(0, 32) || r', pLen + 128 + keyLen), pLen + 128 + keyLen)$ .
3. Parse  $h' = t' || k''$ , where the bit length of  $t'$  is  $pLen + 128$ , the bit length of  $k''$  is  $keyLen$ .
4. Let  $\alpha' = \text{BS2IP}(t', pLen + 128) \bmod p$ .
5. Let  $k' = \text{BS2OSP}(k'', keyLen)$ .
6. Output  $(\alpha', k')$ .

## 8 Auxiliary techniques

This section gives several examples of the techniques that support the functions described in this document.

### 8.1 Hash functions

One hash function is recommended for the encoding methods in this document: SHA-1.

#### 8.1.1 SHA-1

SHA-1 is defined in FIPS PUB 160-1 [1]. The output length of SHA-1 is 160 bits, and the operation block size is 512 bits.

### 8.2 Key derivation functions

One mask generation function is recommended as a key derivation function for the encoding methods in this document: MGF1 [2].

MGF1 is also called KDF1 in [5].

### 8.2.1 MGF1

MGF1 is a mask generation function based on a hash function.

MGF1( $M, l$ ) is defined as follows:

<b>Options:</b>	<i>Hash</i>	hash function
	<i>hashLen</i>	length in bits of the hash function output
<b>Input:</b>	$M$	seed from which mask is generated, an octet string
	$l$	intended bit length of the mask
<b>Output:</b>	<i>mask</i>	mask, an octet string of length $\left\lceil \frac{l}{8} \right\rceil$ octets
<b>Errors:</b>	“invalid”	
<b>Steps:</b>		

1. Let  $l_0$  be the bit length of  $M$ . If  $l_0 + 32$  is greater than the input limitation for the hash function, assert “invalid” and stop.

2. Let  $cThreshold = \left\lceil \frac{l}{hashLen} \right\rceil$ .

3. Let  $M'$  be the empty octet string.

4. Let  $counter = 0$ .

(a) Convert the integer  $counter$  to an octet string of length 32 bits:

$$C = \text{I2OSP}(counter, 32).$$

(b) Concatenate  $M$  and  $C$ , and then apply the hash function to it to produce a hash value:

$$H = \text{Hash}(M \parallel C).$$

(c) Concatenate  $M'$  and  $H$  to the octet string  $M'$ :

$$M' = M' \parallel H.$$

(d) Let  $counter = counter + 1$ . If  $counter < cThreshold$ , then go back to step 4a.

5. Let  $mask$  be the leftmost  $\left\lceil \frac{l}{8} \right\rceil$  octets of the octet string  $M'$ :

$$M'_0 M'_1 \cdots M'_{\lceil l/8 \rceil - 1}.$$

6. Output  $mask$ .

## References

- [1] FIPS PUB 186-2, “Digital Signature Standard (DSS),” U.S. Department of Commerce / National Institute of Standards and Technology, January 27, 2000.
- [2] IEEE Std P1363-2000, “Standard Specifications for Public Key Cryptography: Additional Techniques,” draft version 8, IEEE, April 27, 2001.

- [3] RSA Laboratories, “PKCS #1 v2.1: RSA Encryption Standard,” draft 2, January 5, 2001.
- [4] Fujisaki, E. and Okamoto, T.: “Secure Integration of Asymmetric and Symmetric Encryption Schemes,” Proc. of Crypto’99, Springer-Verlag, LNCS 1666, pp. 535–554 (1999).
- [5] Victor Shoup “A Proposal for an ISO Standard for Public Key Encryption,” <http://shoup.net/papers/iso-2.pdf>.

## A Design principles

PSEC-KEM is the key encapsulation mechanism[5] (called PSEC-2’), which is improved from the the encryption scheme PSEC-2, which is converted from the elliptic curve ElGamal encryption function (encryption primitive) using the conversion in [4] in order to have high level security.

PSEC-KEM takes over advantages of elliptic curve encryption function (elliptic curve ElGamal encryption function) which has various characteristics practically. PSEC-KEM is the encryption scheme which is proved to have the highest security level(semantically secure against adaptive chosen ciphertext attacks).

## B Security requirements of parameters

Security requirements of PSEC-KEM parameters are the following:

$$\begin{aligned} pLen &\geq 160 \\ hLen &\geq 128 \end{aligned}$$

## C Recommendation values of parameters

Recommendation values of PSEC-KEM parameters are the following:

$$\begin{aligned} pLen &= 160 \\ KDF &= \text{MGF1 (SHA-1)} \\ hLen &= 160 \\ R &= \text{Compressed} \\ keyLen &= 256 \end{aligned}$$

## D Fast implementation techniques

We have several implementation techniques for fast computations of PSEC-KEM:

- Using methods for ever-known fast elliptic curve arithmetic.
- Using the pre-computation results for private key, public key, and system parameters.

## E Useful information for implementations

- See SECG about parameters and implementations of elliptic curve cryptosystems, (<http://www.secg.org/>)
- We can implement PSEC-KEM to use some multi-precision library, such as GMP. (<http://www.swox.com/gmp/>)

## F Information about versions

There are four types of PSEC: PSEC-1, PSEC-2, PSEC-3, and PSEC-KEM. Among them, PSEC-2 and PSEC-3 are hybrid encryption schemes. Although PSEC-KEM is a key encapsulation mechanism, we can also utilize it as a hybrid encryption scheme like ISO draft [5] Section 2.7.

We have submitted PSEC specification to four standards, which described below:

(a) ISO draft (PSEC-KEM)[5]

Based on PSEC-2, PSEC-2' which is specified to have a function of key agreement is mentioned in current draft. We will change the name to PSEC-KEM. PSEC-KEM has no compatibility with PSEC-2.

The recommended parameters has not decided yet.

(b) NESSIE draft

In our submission in 2000, the standard includes the primitive of PSEC-1,2,3 only, and does not include any encoding methods.

We plan that we will withdraw PSEC-1,3, and re-submit PSEC-2 as what has a full compatibility with PSEC-KEM in ISO draft. The recommended parameters will be the same to those of CRYPTREC 2001.

(c) CRYPTREC 2000

In our submission in 2000, the standard includes the primitive of PSEC-1,2,3 only, and does not include any encoding methods.

The recommended parameters are  $pLen \leq 160, hLen \leq 128$ .

(d) CRYPTREC 2001 We withdraw PSEC-1,3, and change the name of PSEC-2 in 2000 to PSEC-KEM, and re-submit what is changed to have a full compatibility with PSEC-KEM in ISO draft.

The recommended parameters are  $pLen \leq 160, hLen \leq 128$ .

### F.1 Reasons for changing recommended parameters

There is no change about recommended parameters.

### F.2 User's disadvantage without compatibility

As the encoding methods of PSEC-1, PSEC-2, PSEC-3, and PSEC-KEM are different, they are all different as a scheme. Thus interoperability between them is not satisfied.