PSEC-KEM Specification

NTT Information Sharing Platform Laboratories, NTT Corporation

October 12, 2001

Contents

1 Introduction

This document provides a specification for implementing PSEC-KEM, which is a key encapsulation mechanism(KEM). We can utilize the mechanism for realizing key agreement schemes. This document covers the following issues:

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

This specification is compatible with the PSEC-KEM in ISO/IEC JTC1/SC27 draft [1]. For the usage of KEM in the hybrid encryption applications, see [1].

1.1 Overview

This document is organized 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 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 of the auxiliary functions used in this document.

2 Notation

N	the set of natural integers
$a := b$	give variable a the value of expression b
$(a_1a_0)_{16}$	radix-16 positional notation $a_0 + 16 \cdot a_1$
$\mathbb{F}_{q^m}$	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
$\{0,1\}^*$	$\bigcup_{i=0}^{\infty}\{0,1\}^i$
$\{0,1,\cdots,255\}^i$	the set of all octet strings of length i
$\{0,1,\cdots,255\}^*$	$\bigcup_{i=0}^{\infty}\{0,1,\cdots,255\}^i$
$\|\|$	a concatenation operator for two bit strings or a concatenation operator for two octet strings, for example, $(0,1,0,0)\,\|\|\,(1,1,0) = (0,1,0,0,1,1,0)$ for bit strings, $(4,3)\,\|\|\,(6,2) = (4,3,6,2)$ 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 nonnegative integer b which satisfies $m \mid (b-a)$ for $a, m \in N$

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.

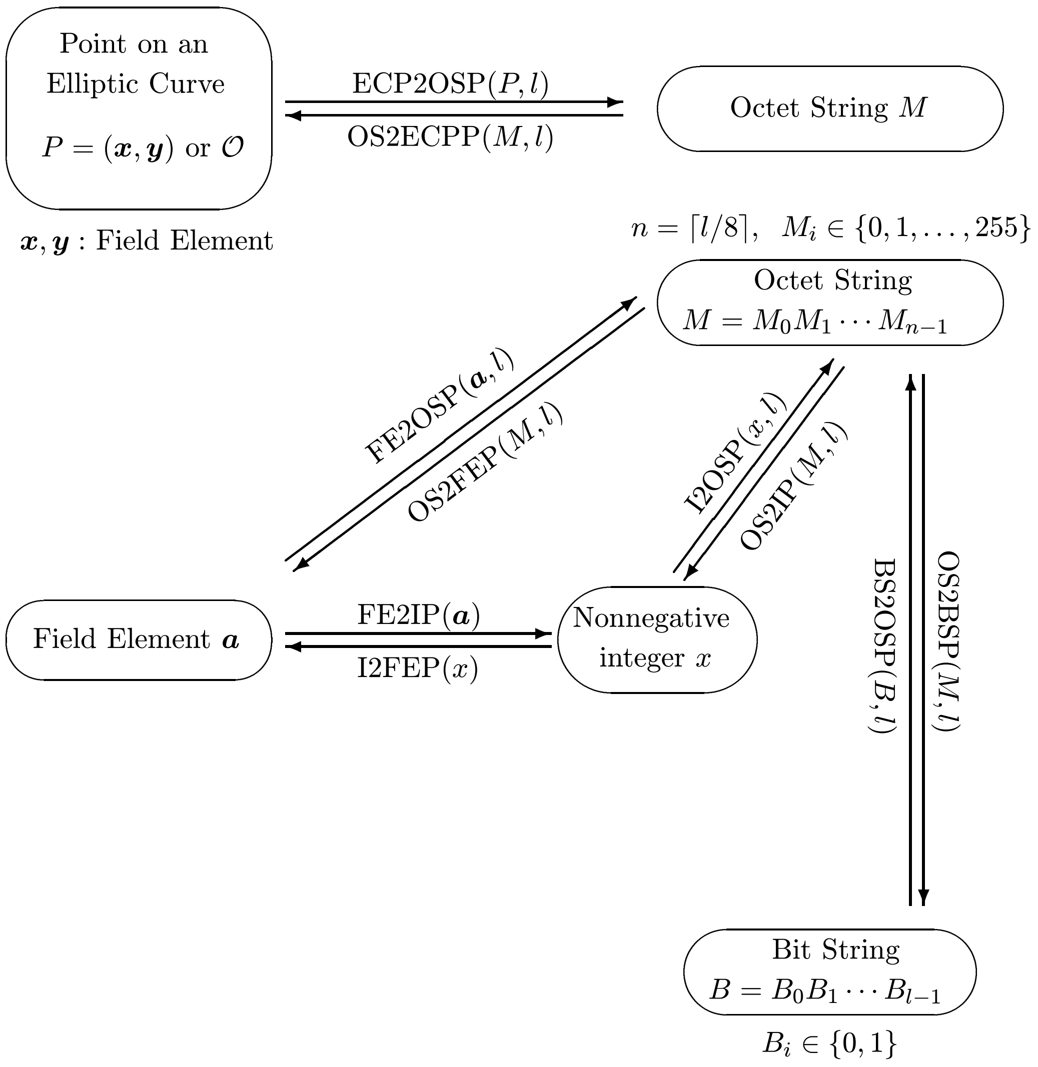
Point on an Elliptic Curve
$P = (\boldsymbol{x}, \boldsymbol{y})$ or $\mathcal{O}$
$\boldsymbol{x}, \boldsymbol{y}$: Field Element
ECP2OSP(P, l)
OS2ECPP(M, l)
Octet String M
$n = \lceil l/8 \rceil, \;\; M_i \in \{0, 1, \ldots, 255\}$
Octet String
$M = M_0 M_1 \cdots M_{n-1}$
FE2OSP$(\boldsymbol{a}, l)$
OS2FEP(M, l)
I2OSP(x, l)
OS2IP(M, l)
BS2OSP(B, l)
OS2BSP(M, l)
Field Element $\boldsymbol{a}$
FE2IP$(\boldsymbol{a})$
I2FEP(x)
Nonnegative integer x
Bit String
$B = B_0 B_1 \cdots B_{l-1}$
$B_i \in \{0, 1\}$

Figure 1: Converting between data types

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, 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 $B = B_0 B_1 \ldots B_{l-1}$ to an octet string $M = M_0 M_1 \ldots M_{n-1}$ as follows:

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

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

2. Set the leftmost $8n-l$ bits in M_0 to 0's, and the rightmost $l+8-8n$ bits to $B_0 B_1 \ldots B_{l+7-8n}$.
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. Formally, the conversion routine, 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 $M = M_0 M_1 \ldots M_{n-1}$ to a bit string $B = B_0 B_1 \ldots B_{l-1}$ as follows:

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

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

2. Ignore the leftmost $8n - l$ bits of M_0, and set $B_0 B_1 \ldots B_{l+7-8n}$ 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 and then convert the resulting bit string to an octet string. Formally, the conversion routine, 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, $x_i \in \{0, \cdots, 255\}$ such that
$$x = x_{n-1}2^{8(n-1)} + x_{n-2}2^{8(n-2)} + \cdots + x_1 2^8 + x_0.$$
3. For $0 \leq i \leq n-1$, set $M_i := x_{n-1-i}$, and let
$$M := M_0 M_1 \ldots M_{n-1}.$$
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, 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 $M = M_0 M_1 \ldots M_{n-1}$ to an integer, x, as follows:

1. View each M_i as an integer in $\{0, \ldots, 255\}$, and compute
$$x := \sum_{i=0}^{n-1} 2^{8(n-1-i)} M_i \mod 2^l.$$
2. Output x.

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, FE2IP$(\boldsymbol{a})$, 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:

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

Output:

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

Steps:

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

```
if $m = 1$:
    Field element $\boldsymbol{a}$ must be represented as an integer in $\{0, \ldots, q^m - 1\}$.
    1: Let $x := \boldsymbol{a}$.
    2: Output $x$.
if $m > 1$:
    Field element $\boldsymbol{a}$ must be represented as a polynomial of at most $(m-1)$-th degree with coefficients in $\{0, \ldots, q-1\}$. Let $\beta$ be the variable of the polynomial.
    1: Determine the coefficients $a_i \in \{0, \ldots, q-1\}$ for $i \in \{0, \ldots, m-1\}$ that satisfy
           $\boldsymbol{a} = \sum_{i=0}^{m-1} a_i \beta^i.$
    2: Compute
           $x := \sum_{i=0}^{m-1} a_i q^i.$
    3: Output $x$.
```

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, 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 $\{0, \ldots, q^m - 1\}$

Output:

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

Steps:

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

```
if $m = 1$:
    A field element of $\mathbb{F}_{q^m}$ must be represented as an integer in $\{0, \ldots, q^m - 1\}$.
    1: Let $\boldsymbol{a} := x$.
    2: Output $\boldsymbol{a}$.
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 $\{0, \ldots, q-1\}$. Let $\beta$ be the variable of the polynomial.
    1: Expand $x$ into it's radix $q$ representation $x_i \in \{0, \ldots, q-1\}$ for $i \in \{0, \ldots, m-1\}$ that satisfies
           $$x = \sum_{i=0}^{m-1} x_i q^i.$$
    2: Compute
           $$\boldsymbol{a} := \sum_{i=0}^{m-1} x_i \beta^i.$$
    3: Output $\boldsymbol{a}$.
```

3.7 FieldElement-to-OctetString Conversion(FE2OSP)

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

Input:

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

Output:

M	:	an octet string

Steps:

```
1: Let
       $$M := \text{I2OSP}(\text{FE2IP}(\boldsymbol{a}), l).$$
2: Output $M$.
```

3.8 OctetString-to-FieldElement Conversion(OS2FEP)

The conversion routine, OS2FEP(M, l), is specified as follows:

Input:

M	:	an octet string
l	:	an integer

Output:

$\boldsymbol{a}$	:	a field element

Steps:

```
1: Let
        $$\boldsymbol{a} := \mathrm{I2FEP}(\mathrm{OS2IP}(M, l)).$$
2: Output $\boldsymbol{a}$.
```

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, 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 $M = M_0 M_1 \dots M_{n-1}$ as follows:
  1. If $P = \mathcal{O}$, output $M := (00)_{16}$ .
  2. If $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$ and $R =$ Compressed, proceed as follows:
    2.1. Set octet string $X :=$ FE2OSP$(\boldsymbol{x}, l)$.
    2.2. Derive from $\boldsymbol{y}$ 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 $\boldsymbol{y} =$
             $y_{m-1}\beta^{m-1} + \cdots + y_1\beta + y_0$.
```

```
        2.2.2. If $q = 2$ ,set $\tilde{y} := 0$ if $\boldsymbol{x} = \mathbf{0}$, otherwise compute
               $\boldsymbol{z} = z_{m-1}\beta^{m-1} + \cdots + z_1\beta + z_0$ such that $\boldsymbol{z} = \boldsymbol{y}\boldsymbol{x}^{-1}$ 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 $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$ and $R =$ Uncompressed, proceed as follows:
    3.1. Set octet string $X :=$ FE2OSP$(\boldsymbol{x}, l)$.
    3.2. Set octet string $Y :=$ FE2OSP$(\boldsymbol{y}, l)$.
    3.3. Output $M := (04)_{16}\|X\|Y$.
4. If $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$ and $R =$ Hybrid, proceed as follows:
    4.1. Set octet string $X :=$ FE2OSP$(\boldsymbol{x}, l)$.
    4.2. Set octet string $Y :=$ FE2OSP$(\boldsymbol{y}, l)$.
    4.3. Derive from $\boldsymbol{y}$ 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 $\boldsymbol{y} =$
               $y_{m-1}\beta^{m-1} + \cdots + y_1\beta + y_0$.
        4.3.2. If $q = 2$ ,set $\tilde{y} := 0$ if $\boldsymbol{x} = \mathbf{0}$, otherwise compute
               $\boldsymbol{z} = z_{m-1}\beta^{m-1} + \cdots + z_1\beta + z_0$ such that $\boldsymbol{z} = \boldsymbol{y}\boldsymbol{x}^{-1}$ 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(OS2ECPP)

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, OS2ECPP(M, l), is specified as follows:

Option:

E	:	an elliptic curve parameter

Input:

M	:	an octet string that 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 a point P on E 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 $\boldsymbol{x} := \mathrm{OS2FEP}(X, l)$.
   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 $\boldsymbol{x}$ and $\tilde{y}$ elliptic curve point $P := (\boldsymbol{x}, \boldsymbol{y})$, where:
      2.4.1. If $q$ is an odd number, compute the field element $\boldsymbol{w} := \boldsymbol{x}^3 + \boldsymbol{a}\boldsymbol{x} + \boldsymbol{b}$ ,and compute a square root $\boldsymbol{\gamma}$ of $\boldsymbol{w}$ in $\mathbb{F}_{q^m}$. Assert "invalid" and stop if there are no square roots in $\mathbb{F}_{q^m}$ , otherwise set $\boldsymbol{y} := \boldsymbol{\gamma}$ if $\beta_0 \equiv \tilde{y} \bmod 2$, and set $\boldsymbol{y} := -\boldsymbol{\gamma}$ if $\gamma_0 \not\equiv \tilde{y} \bmod 2$, where $\boldsymbol{\gamma} = \gamma_{m-1}\beta^{m-1} + \cdots + \gamma_1\beta + \gamma_0$.
      2.4.2. If $q = 2$ and $\boldsymbol{x} = \mathbf{0}$, set $\boldsymbol{y} := \boldsymbol{b}^{2^{m-1}}$ in $\mathbb{F}_{q^m}$.
      2.4.3. If $q = 2$ and $\boldsymbol{x} \neq 0$, compute the field element $\boldsymbol{\gamma} := \boldsymbol{x} + \boldsymbol{a} + \boldsymbol{b}\boldsymbol{x}^{-2}$ in $\mathbb{F}_{q^m}$, and find an element $\boldsymbol{z} = z_{m-1}\beta^{m-1} + \cdots + z_1\beta + z_0$ such that $\boldsymbol{z}^2 + \boldsymbol{z} = \boldsymbol{\gamma}$ in $\mathbb{F}_{q^m}$. Assert "invalid" and stop if no such $\boldsymbol{z}$ exists, otherwise set $\boldsymbol{y} := \boldsymbol{x}\boldsymbol{z}$ in $\mathbb{F}_{q^m}$ if $z_0 = \tilde{y}$, and set $\boldsymbol{y} := \boldsymbol{x}(\boldsymbol{z} + \mathbf{1})$ in $\mathbb{F}_{q^m}$ if $z_0 \neq \tilde{y}$.
   2.5. Output $P := (\boldsymbol{x}, \boldsymbol{y})$.
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 $\boldsymbol{x} := \mathrm{OS2FEP}(X, l)$.
   3.4. Set $\boldsymbol{y} := \mathrm{OS2FEP}(Y, l)$.
   3.5. If $P := (\boldsymbol{x}, \boldsymbol{y})$ does not satisfy the defining equation of elliptic curve $E$, then assert "invalid" and stop.
   3.6. Output $P := (\boldsymbol{x}, \boldsymbol{y})$.
```

4 Key types

In this section, two types of keys are defined: PSEC private key and PSEC public key, both of 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 the 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 $(q, m, f(\boldsymbol{\beta}), \boldsymbol{a}, \boldsymbol{b}, P, p, pLen, qmLen)$, where the components have the following meanings:

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

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

- 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, $W = sP$ holds, where s is a PSEC private key as decribed in Section 4.1.

Note :
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 $s \in \{0, \cdots, p-1\}$.
2. Let $W := sP$.
3. Output $PK = (E, W, KDF, hLen)$ and s.

5.2 EP-PSEC

EP-PSEC(PK, α) is defined as follows:

Input :	PK	PSEC public key
	α	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 an 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.

PSEC-KEM consists of two operations.

- The encryption operation ES-PSEC-KEM-Encrypt(PK) that takes as input public key PK and outputs ciphertext/key pair(c_0, k).
- The decryption operation ES-PSEC-KEM-Decrypt(PK, s, c_0) that takes as input public key PK, private key s and ciphertext c_0, and outputs key k.

6.1 ES-PSEC-KEM

6.1.1 Encryption operation

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 $(\alpha, k, r) :=$ EME-PSEC-KEM-A(PK). (See Section 7.1.1.)
2. Let $(Q, C_1) :=$ EP-PSEC(PK, α). (See Section 5.2.)
3. Let $c_0 :=$ EME-PSEC-KEM-B(PK, Q, C_1, r). (See Section 7.1.2.)
4. Output (c_0, k).

6.1.2 Decryption operation

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 $(C_1, c_2, g) :=$ 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 $Q' :=$ DP-PSEC(PK, C_1, s). (See Section 5.3.)
3. Let (α', k'):= EME-PSEC-KEM-D(PK, c_2, g, Q'). (See Section 7.1.4.)
4. Check $C_1 :=$ DP-PSEC(PK, P, α'). (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 :	α	a nonnegative integer, $0 \leq \alpha < p$
	k	an octet string
	r	an octet string
Steps :		

1. Generate a random octet string $r \in \{0, \cdots, 255\}^{\lceil hLen/8 \rceil}$.
2. Let $H :=$ OS2BSP$(KDF(\text{I2OSP}(0, 32) \,||\, r, pLen{+}128{+}keyLen), pLen{+}128 + keyLen)$.

3. Parse $H = t \,||\, k'$, where the bit length of t is $pLen + 128$; the bit length of k' is $keyLen$.
4. Let $\alpha :=$ BS2IP$(t, pLen + 128) \bmod p$.
5. Let $k :=$ BS2OSP$(k', keyLen)$.
6. Output (α, 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(I2OSP$(1, 32) \,||\,$ ECP2OSP$(C_1, qmLen) \,||\,$ ECP2OSP$(Q, qmLen), hLen)$.
2. Let $c_0 :=$ ECP2OSP$(C_1, qmLen) \,||\, c_2$.
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 $C_1 :=$ OS2ECPP$(g, qmLen)$.
   If OS2ECPP 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 :	α'	a nonnegative integer, $0 \leq \alpha' < p$
	k'	an octet string

Steps :

1. Let $r' := c_2 \oplus KDF(\mathrm{I2OSP}(1, 32) \,||\, g \,||\, \mathrm{ECP2OSP}(Q', qmLen), hLen)$.
2. Let $h' := \mathrm{OS2BSP}(KDF(\mathrm{I2OSP}(0, 32) \,||\, r', pLen{+}128{+}keyLen), pLen + 128 + keyLen)$.
3. Parse $h' = t' \,||\, k''$, where the bit length of t' is $pLen + 128$; the bit length of k'' is $keyLen$.
4. Let $\alpha' := \mathrm{BS2IP}(t', pLen + 128) \bmod p$.
5. Let $k' := \mathrm{BS2OSP}(k'', keyLen)$.
6. Output (α', 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 180-1 [2]. 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 [3].

MGF1 is also called KDF1 in [1].

8.2.1 MGF1

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

MGF1(M, l) is defined as follows:

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

Steps:

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:
   $$C := \mathrm{I2OSP}(counter, 32).$$
   (b) Concatenate M and C, and apply the hash function to the result to produce a hash value:
   $$H := Hash(M \mid\mid C).$$
   (c) Concatenate M' and H to octet string M':
   $$M' := M' \mid\mid H.$$
   (d) Let $counter := counter + 1$. If $counter < cThreshold$, 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] V. Shoup. "A Proposal for an ISO Standard for Public Key Encryption (v.2.0)," ISO/IEC JTC1/SC27, N2918, http://shoup.net/papers/, 2001 Sep.

[2] FIPS PUB 180-1, "Secure Hash Standard (SHS)," U.S. Department of Commerce / National Institute of Standards and Technology, April 17, 1995.

[3] RSA Laboratories, "PKCS #1 v2.1: RSA Encryption Standard," draft 2, January 5, 2001.

A Security requirements of parameters

Security requirements of PSEC-KEM parameters are the following:

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

B Recommended values of parameters

Recommended values of PSEC-KEM parameters are the following:

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