A Proposal of PSEC-KEM to IEEE P1363

NTT Information Sharing Platform Laboratories
Nippon Telegraph and Telephone Corporation

August 24, 2006

1 Description of the Proposed Scheme

The description of the proposed scheme PSEC-KEM is provided in the attached document 1.

2 Attributes and Advantages

1. PSEC-KEM is provably secure under the computational Diffie-Hellman assumption over elliptic curves in the random oracle model.
2. PSEC-KEM is efficient. The efficiency assessment is provided in the attached document 2.
3. PSEC-KEM bases on the Key Encapsulation Mechanism (KEM). Together with the Data Encapsulation Mechanism (DEM), KEM is accepted in ISO and NESSIE standards.

3 Security Assessment

The security assessment is also provided in the attached document 2.

4 Known Limitations

Canetti *et al.* [1] have demonstrated that it is possible to devise cryptographic protocols which are provably secure in the random oracle model but for which no complexity assumption property instantiates the random-oracle-modeled hash function. However, the examples they used to make the random oracle model paradigm fail were very contrived, so the concerns induced by these examples do not appear to apply to any of the concrete practical schemes that have been proven secure in the random oracle model.

5 Intellectual Property Statement

NTT has filed patent applications on the techniques used in this contribution. NTT will license any resulting patent in a reasonable and non-discriminatory fashion. A letter to this effect will be provided.

References

[1] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracles Methodology, Revisited. In *Proc. of the 30th STOC*, pages 209–218. ACM Press, New York, 1998.

A Proposal of PSEC-KEM to IEEE P1363 Attached Document 1: PSEC-KEM Specification

NTT Information Sharing Platform Laboratories,
Nippon Telegraph and Telephone Corporation

August 24, 2006

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
- key encapsulation mechanisms: ES-PSEC-KEM

This specification is compatible with the PSEC-KEM in ISO/IEC 18033-2 [1]. For the usage of KEM in the hybrid encryption applications, see [1].

2 Notation

bit	one of the two symbols 0 or 1.
bit string	an ordered sequence of bits.
octet	a bit string of length 8.
octet string	an ordered sequence of octets.
$\boldsymbol{R}$	the set of real numbers.
$\boldsymbol{Z}$	the set of integers.
$\boldsymbol{N}$	the set of positive integers.
$a := b$	assign b to a.
$\mathbb{F}_{q^m}$	a finite field with q^m elements, where q is a prime.
$\mathcal{O}$	a point at infinity on an elliptic curve
$\langle B_0, B_1, \ldots, B_{i-1} \rangle$	a bit string of length i, for example, $\langle 0, 1, 0, 0 \rangle$.
$\langle M_0, M_1, \ldots, M_{i-1} \rangle$	an octet string of length i, for example, $\langle 170, 255, 0 \rangle$.
$\|$	a concatenation operator for two bit strings or for two octet strings, for example, $\langle 0, 1, 0, 0 \rangle \,\|\, \langle 1, 1, 0 \rangle = \langle 0, 1, 0, 0, 1, 1, 0 \rangle$ for bit strings, $\langle 170, 255 \rangle \,\|\, \langle 0, 20 \rangle = \langle 170, 255, 0, 20 \rangle$ for octet strings. The operator is often omitted.
$\oplus$	the bit-wise exclusive-or operation
$\lceil y \rceil$	for $y \in \boldsymbol{R}$, the least integer greater than or equal to y.
$\lfloor y \rfloor$	for $y \in \boldsymbol{R}$, the greatest integer less than or equal to y.
$\mathrm{GCD}(a, b)$	for $a \in \boldsymbol{N}$, $b \in \boldsymbol{N}$, the greatest common divisor of a and b.
$a \mid b$	for $a \in \boldsymbol{Z}$, $b \in \boldsymbol{Z}$, a divides b.
$a \bmod m$	for $a \in \boldsymbol{Z}$, $m \in \boldsymbol{N}$, the least nonnegative integer b which satisfies $m \mid (a - b)$.
$a^{-1} \bmod m$	for $a \in \boldsymbol{Z}$, $m \in \boldsymbol{N}$, the least nonnegative integer b which satisfies $ab \bmod m = 1$.

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 Integer-to-BitString Conversion (I2BSP)

Integers should be converted to bit strings as described in this section. Informally, the idea is to represent the integer in binary. Formally, the conversion routine, $\mathrm{I2BSP}(x, l)$, is specified as follows:

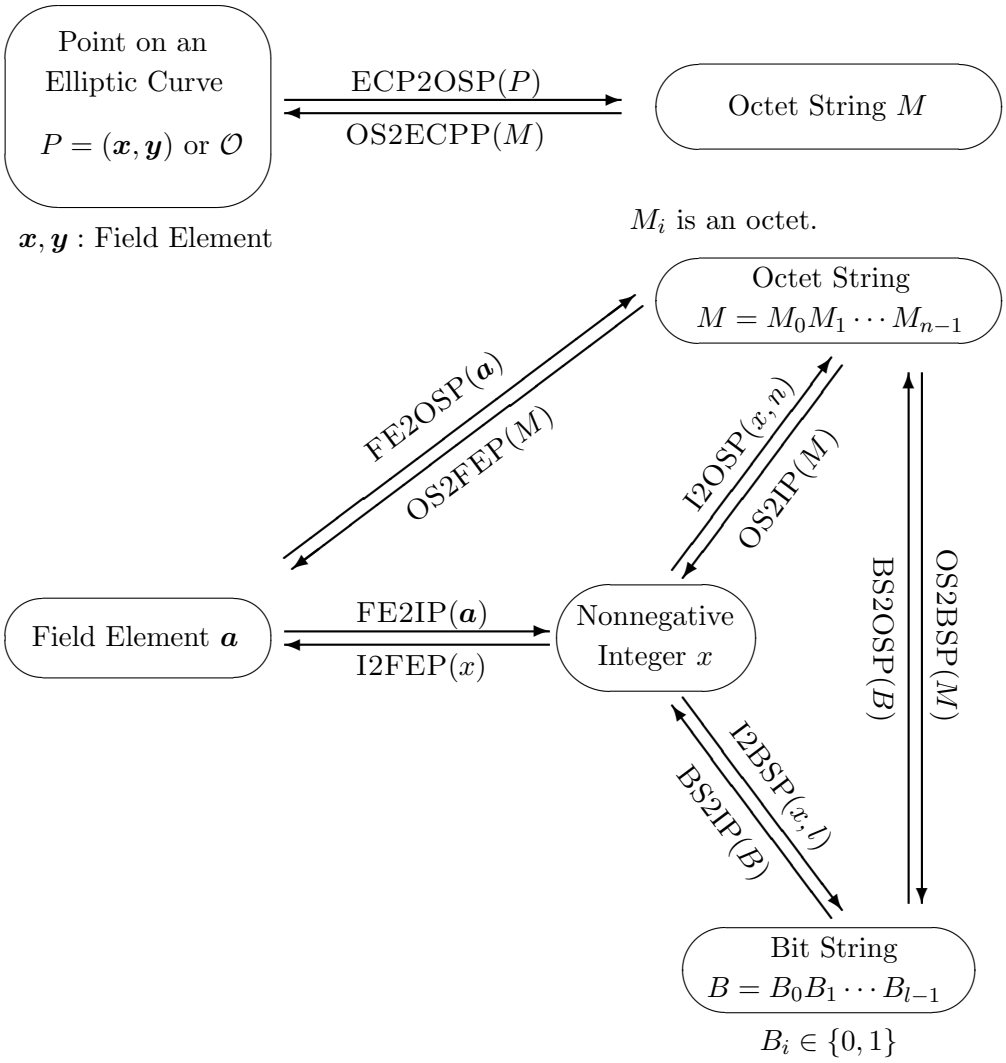

Figure 1: Conversion between data types

Input: x an integer to be converted, a nonnegative integer
l the bit length of the output, a nonnegative integer
Output: B a bit string of length l bits
Errors: `INVALID`
Steps:

1. If $l = 0$, output an empty bit string and stop.
2. If $x \geq 2^l$, assert `INVALID` and stop.
3. Determine the x's base-2 representation, $x_i \in \{0, 1\}$ such that
$$x = x_{l-1}2^{l-1} + x_{l-2}2^{l-2} + \cdots + x_1 2 + x_0.$$
4. For $0 \leq i \leq l - 1$, set $B_i := x_{l-1-i}$, and let
$$B := B_0 B_1 \cdots B_{l-1}.$$
5. Output B.

3.2 BitString-to-Integer Conversion (BS2IP)

Bit strings should be converted to integers as described in this section. Informally, the idea is simply to view the bit string as the base-2 representation of the integer. Formally, the conversion routine, BS2IP(B), is specified as follows:

Input: B a bit string of length l to be converted
Output: x a nonnegative integer
Steps:

Convert $B = B_0B_1 \cdots B_{l-1}$ to an integer x as follows:

1. If $l = 0$, output 0 and stop.
2. View each B_i as an integer in $\{0, 1\}$, set $x_i := B_i$ for $0 \leq i \leq l - 1$, and let
$$x := \sum_{i=0}^{l-1} 2^{(l-1-i)} x_i.$$
3. Output x.

3.3 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), is specified as follows:

Input: B a bit string of length l to be converted
Output: M an octet string of length $n = \left\lceil \frac{l}{8} \right\rceil$ octets
Steps:

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

1. If $l = 0$, output an empty octet string and stop.
2. For $0 < i \leq n - 1$, let:
$$M_i := B_{l-8-8(n-1-i)}B_{l-7-8(n-1-i)} \cdots B_{l-1-8(n-1-i)}.$$
3. Set
$$M_0 := \begin{cases} B_0B_1 \cdots B_7 & (8n - l = 0) \\ ZB_0B_1 \cdots B_{l+7-8n} & (8n - l \neq 0) \end{cases},$$
where Z is an all zero bit string of length $8n - l$ ($Z = \langle 0, 0, \ldots, 0 \rangle$).
4. Output M.

3.4 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), is specified as follows:

Input: M an octet string of length n octets to be converted
Output: B a bit string of length l bits
Steps:

Convert the octet string $M = M_0 M_1 \cdots M_{n-1}$ to a bit string $B = B_0 B_1 \cdots B_{l-1}$ as follows:

1. If $l = 0$, output an empty bit string and stop.
2. For $0 < i \leq n - 1$, set:

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

3. For $0 \leq j \leq l + 7 - 8n$, set $B_j := Z_{j+8n-l}$, where $Z_0 Z_1 \cdots Z_7 := M_0$.
4. Output B.

3.5 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, n), is specified as follows:

Input: x an integer to be converted, a nonnegative integer
n the octet length of output.
Output: M an octet string of length n octets
Errors: `INVALID`
Steps:

$$\text{BS2OSP}(\text{I2BSP}(x, 8n))$$

3.6 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), is specified as follows:

Input: M an octet string of length n octets to be converted
Output: x a nonnegative integer
Steps:

Convert $M = M_0 M_1 \cdots M_{n-1}$ to an integer, x, as follows:

$$\text{BS2IP}(\text{OS2BSP}(M))$$

3.7 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 $\{0, \ldots, q-1\}$.

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 β 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.

3.8 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 $\{0, \ldots, q-1\}$.

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 β 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 $\boldsymbol{a}$.

3.9 FieldElement-to-OctetString Conversion (FE2OSP)

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

Input: $\boldsymbol{a}$ a field element
Output: M an octet string
Steps:

1. Let

$$M := \text{I2OSP}(\text{FE2IP}(\boldsymbol{a})).$$

2. Output M.

3.10 OctetString-to-FieldElement Conversion (OS2FEP)

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

Input: M an octet string
Output: $\boldsymbol{a}$ a field element
Steps:

1. Let

$$\boldsymbol{a} := \text{I2FEP}(\text{OS2IP}(M)).$$

2. Output $\boldsymbol{a}$.

3.11 EllipticCurvePoint-to-OctetString Conversion (ECP2OSP)

Elliptic curve points should be converted to octet strings as described in this section. Informally the idea is that, if point compression is being used, 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), is specified as follows:

System parameters: E an elliptic curve parameter
R Compressed, Uncompressed, or Hybrid
Input: P a point on an elliptic curve over $\mathbb{F}_{q^m}$
Output: M an octet string of length n

$$\text{where } \begin{cases} n = 1 & \text{if } P = \mathcal{O}, \\ n = \left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil + 1 & \text{if } P \neq \mathcal{O} \text{ and } R \text{ is Compressed}, \\ n = 2\left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil + 1 & \text{if } P \neq \mathcal{O} \text{ and } R \text{ is Uncompressed or Hybrid.} \end{cases}$$

Steps:

Convert P to an octet string $M = M_0 M_1 \cdots M_{n-1}$ as follows:

1. If $P = \mathcal{O}$, output $M := \text{I2OSP}(00, 1)$.
2. If $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$ and $R =$ Compressed, proceed as follows:
   2.1. Set octet string $X := \text{FE2OSP}(\boldsymbol{x})$.
   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 $\tilde{y} := 0$ if $\boldsymbol{y} = \boldsymbol{0}$, set $\tilde{y} := y_i \bmod 2$ if $\boldsymbol{y} \neq \boldsymbol{0}$, where $\boldsymbol{y} = y_{m-1}\beta^{m-1} + \cdots + y_1\beta + y_0$, and i is the smallest integer such that $y_i \neq 0$.
      2.2.2. If $q = 2$, set $\tilde{y} := 0$ if $\boldsymbol{x} = \boldsymbol{0}$, 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 $\text{I2OSP}(02, 1)$ to the single octet L. If $\tilde{y} = 1$, assign the value $\text{I2OSP}(03, 1)$ 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 $X := \text{FE2OSP}(\boldsymbol{x})$.
   3.2. Set octet string $Y := \text{FE2OSP}(\boldsymbol{y})$.
   3.3. Output $M := \text{I2OSP}(04, 1) \,\|\, X \,\|\, Y$.
4. If $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$ and $R =$ Hybrid, proceed as follows:
   4.1. Set octet string $X := \text{FE2OSP}(\boldsymbol{x})$.
   4.2. Set octet string $Y := \text{FE2OSP}(\boldsymbol{y})$.
   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 $\tilde{y} := 0$ if $\boldsymbol{y} = \boldsymbol{0}$, set $\tilde{y} := y_i \bmod 2$ if $\boldsymbol{y} \neq \boldsymbol{0}$, where $\boldsymbol{y} = y_{m-1}\beta^{m-1} + \cdots + y_1\beta + y_0$, and i is the smallest integer such that $y_i \neq 0$.

4.3.2. If $q = 2$, set $\tilde{y} := 0$ if $\boldsymbol{x} = \boldsymbol{0}$, 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 I2OSP$(06, 1)$ to the single octet L. If $\tilde{y} = 1$, assign the value I2OSP$(07, 1)$ to the single octet L.

4.5. Output $M := L \,\|\, X \,\|\, Y$.

3.12 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), is specified as follows:

System parameters:	E	an elliptic curve parameter
Input:	M	an octet string that is either the single octet, an octet string of length $n = \left\lceil \frac{\log_2(q^m - 1)}{8} \right\rceil + 1$, or an octet string of length $n = 2\left\lceil \frac{\log_2(q^m - 1)}{8} \right\rceil + 1$
Output:	P	an elliptic curve point
Errors:	`INVALID`	
Steps:		

Convert M to a point P on E as follows:

1. If $M = $ I2OSP$(00, 1)$, output $P := \mathcal{O}$.
2. If M has length $\left\lceil \frac{\log_2(q^m - 1)}{8} \right\rceil + 1$ octets, proceed as follows:

   2.1. Parse $M = L \,\|\, X$ as a single octet L followed by $\left\lceil \frac{\log_2(q^m - 1)}{8} \right\rceil$ octets X.

   2.2. Set $\boldsymbol{x} := $ OS2FEP(X).

   2.3. If $L = $ I2OSP$(02, 1)$, set $\tilde{y} := 0$, and if $L = $ I2OSP$(03, 1)$, 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 $\mathbb{F}_{q^m}$, set $\boldsymbol{y} = \boldsymbol{0}$ if $\boldsymbol{\gamma} = \boldsymbol{0}$, otherwise set $\boldsymbol{y} := \boldsymbol{\gamma}$ if $\gamma_i \equiv \tilde{y} \bmod 2$, and set $\boldsymbol{y} := -\boldsymbol{\gamma}$ if $\gamma_i \not\equiv \tilde{y} \bmod 2$, where $\boldsymbol{\gamma} = \gamma_{m-1}\beta^{m-1} + \cdots + \gamma_1\beta + \gamma_0$, and i is the smallest integer such that $\gamma_i \neq 0$.

      2.4.2. If $q = 2$ and $\boldsymbol{x} = \boldsymbol{0}$, set $\boldsymbol{y} := \boldsymbol{b}^{2^{m-1}}$ in $\mathbb{F}_{q^m}$.

2.4.3. If $q = 2$ and $\boldsymbol{x} \neq \boldsymbol{0}$, 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 $\boldsymbol{y} := \boldsymbol{x}(\boldsymbol{z} + \boldsymbol{1})$ in $\mathbb{F}_{q^m}$ if $z_0 \neq \tilde{y}$.

2.5. Output $P := (\boldsymbol{x}, \boldsymbol{y})$.

3. If M has length $2\left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil + 1$ octets, proceed as follows:

3.1. Parse $M = L \,\|\, X \,\|\, Y$ as a single octet L followed by $\left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil$ octets X followed by $\left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil$ octets Y.

3.2. Check that $L = \mathrm{I2OSP}(04, 1)$ or $\mathrm{I2OSP}(06, 1)$ or $\mathrm{I2OSP}(07, 1)$. If $L \neq \mathrm{I2OSP}(04, 1)$ or $\mathrm{I2OSP}(06, 1)$ or $\mathrm{I2OSP}(07, 1)$, assert `INVALID` and stop.

3.3. Set $\boldsymbol{x} := \mathrm{OS2FEP}(X)$.

3.4. Set $\boldsymbol{y} := \mathrm{OS2FEP}(Y)$.

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 $P := (\boldsymbol{x}, \boldsymbol{y})$.

3.13 Partial EllipticCurvePoint-to-OctetString Conversion (PECP2OSP)

System parameters:	E	an elliptic curve parameter
Input:	P	a point on an elliptic curve over $\mathbb{F}_{q^m}$
Output:	M	an octet string of length n where $n = \left\lceil \dfrac{\log_2(q^m - 1)}{8} \right\rceil$.

Steps:

Convert P to an octet string $M = M_0 M_1 \cdots M_{n-1}$ as follows:

1. If $P = \mathcal{O}$, output $M := \mathrm{I2OSP}(0, n)$.
2. If $P = (\boldsymbol{x}, \boldsymbol{y}) \neq \mathcal{O}$, output $\mathrm{FE2OSP}(\boldsymbol{x})$.

4 Key types

In this section, two types of keys are defined: PSEC-KEM system parameters, PSEC-KEM private key and PSEC-KEM public key.

4.1 PSEC-KEM system parameters

A PSEC-KEM system parameters are the following value:

- E, an elliptic curve parameter
- *KDF*, the choice from key derivation functions
- *hLen*, a nonnegative integer
- *keyLen*, 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 $\mathbb{F}_q$
- $\boldsymbol{a}$, 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 $\lceil \log_{256} p \rceil$
- $qmLen$, the value of $\lceil \log_{256} q^m \rceil$

4.2 PSEC-KEM private key

A PSEC-KEM private key is the following value:

- s, a nonnegative integer

4.3 PSEC-KEM public key

A PSEC-KEM public key is the following value:

- W, a point on E

Valid PSEC-KEM public key $W = sP$ holds, where s is one of the element in PSEC-KEM private key as decribed in Section 4.2 and W is a PSEC-KEM public key as decribed in Section 4.3.

5 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.

5.1 KGP-PSEC

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

Input:	(E, KDF, $hLen$, $keyLen$)	PSEC-KEM system parameters
Output:	W	PSEC-KEM public key
	s	PSEC-KEM 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 W and s.

5.2 ES-PSEC-KEM

5.2.1 Encryption operation

ES-PSEC-KEM-ENCRYPT(PK) is defined as follows:

Input:	W	PSEC-KEM public key
Output:	c_0	an octet string
	k	an octet string
Assumptions:	public key PK is valid.	

Steps:

1. Generate a random octet string $r \in \{0, \cdots, 255\}^{hLen}$.
2. Let $H := KDF(\text{I2OSP}(0, 4) \,\|\, r, pLen + 16 + keyLen)$.
3. Parse $H = t \,\|\, k'$, where the octetlength of t is $pLen + 16$; the octet length of k' is $keyLen$.
4. Let $\alpha := \text{OS2IP}(t, pLen + 16) \bmod p$.
5. Let $Q := \alpha W$.
6. Let $C_1 := \alpha P$.
7. Let $c_2 := r \oplus KDF(\text{I2OSP}(1, 4) \,\|\, \text{ECP2OSP}(C_1) \,\|\, \text{PECP2OSP}(Q), hLen)$.
8. Let $c_0 := \text{ECP2OSP}(C_1) \,\|\, c_2$.
9. Output (c_0, k').

5.2.2 Decryption operation

ES-PSEC-KEM-DECRYPT(PK, s, c_0) is defined as follows:

Input:	PK	PSEC-KEM public key
	s	PSEC-KEM 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. If the octet length of c_0 is less than or equal to $hLen$, assert `INVALID` and stop.
2. Parse $c_0 = g \,\|\, c_2$, where the octet length of $hLen$.
3. Let $C_1 := \mathrm{OS2ECP}(g)$.
   If OS2ECP asserts `INVALID`, assert `INVALID` and stop.
4. Let $Q := sC_1$.
5. Let $r' := c_2 \oplus KDF(\mathrm{I2OSP}(1,4) \,\|\, g \,\|\, \mathrm{PECP2OSP}(Q'), hLen)$.
6. Let $h' := KDF(\mathrm{I2OSP}(0,8) \,\|\, r', pLen + 16 + keyLen)$.
7. Parse $h' = t' \,\|\, k'$, where the octet length of t' is $pLen + 16$; the octet length of k' is $keyLen$.
8. Let $\alpha' := \mathrm{OS2IP}(t') \bmod p$.
   then assert `INVALID` and stop.
9. Check $C_1 := \alpha' P$.
   If it holds, output k'. Otherwise, assert `INVALID` and stop.

References

[1] ISO/IEC 18033-2 Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric ciphers.

[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 Proposal of PSEC-KEM to IEEE P1363 Attached Document 2: Self Evaluation of PSEC-KEM

NTT Information Sharing Platform Laboratories,
Nippon Telegraph and Telephone Corporation

August 24, 2006

1 Introduction

This document describes security and performance assessment of PSEC-KEM [5]. PSEC-KEM is one of the standardized algorithms of Key Encapsulation Mechanism (KEM) in ISO/IEC 18033-2 and NESSIE. Combined with a DEM (Data Encapsulation Mechanism) primitive, PSEC-KEM can produce a chosen-ciphertext secure asymmetric encryption scheme.

PSEC-KEM is an elliptic curve ElGamal-based key agreement scheme, based on the work of [2] with some modifications [4]. PSEC-KEM is secure in the random oracle model in the sense of secure KEM defined in ISO/IEC 18033-2 [4], assuming the elliptic curve version of computational Diffie-Hellman (EC-CDH) problem is hard.

2 The KEM-DEM Framework

The KEM-DEM framework is introduced by Shoup [3]. Informally speaking, key encapsulation mechanism (KEM) provides a framework of a key agreement scheme in public-key cryptosystems, while date encapsulation mechanism (DEM) defines a symmetric encryption scheme. If both KEM and DEM primitives satisfy the security requirements in [4], a hybrid cipher HC properly obtained from them satisfies the strong security notion of so-called indistinguishability against the adaptive chosen-ciphertext attacks (IND-CCA) for asymmetric encryption.

The security of KEM is defined by inability of an adversary in the chosen-ciphertext attack to distinguish a real key/ciphertext pair (K^*, C^*) from a fake pair $(\tilde{K}, C^*)$. The security of DEM is defined by indistinguishability of an adversary in the "chosen-ciphertext" attack for symmetric encryption; One example of secure DEM is a one-time-pad-then-MAC. For more details about definitions, security notions and composition theorem of these mechanisms, see [4].

3 Assessment of PSEC-KEM

PSEC-KEM is secure in the random oracle model in the sense of secure KEM defined in ISO/IEC 18033-2 [4], assuming the elliptic curve version of computational Diffie-Hellman (EC-CDH) problem is hard.

More precisely, we have the following theorem.

Theorem 3.1 ([4]) *For a given value of the system parameter SeedLen, and for any PSEC-KEM $[t, q]$-adversary A that makes at most q' random oracle queries, we have $\mathbf{Adv}_{\mathrm{PSEC-KEM}}(A) = O(\mathbf{Adv}_{CDH}(A') + (q + q')(\mu^{-1} + 2^{-\mathrm{SeedLen}}))$, where A' is a $CDH[t', O(q + q')]$-adversary, with $t' \sim t$ and μ is the number of elements of the CDH group.*

Here, a CDH adversary A' takes a CDH instance and produces a list of candidate solutions of the CDH problem. If a CDH adversary A' runs in time t and produces a list of at most l group elements, then A' is called a $CDH[t, l]$-adversary. The advantage of A' is denoted by $\mathbf{Adv}_{CDH}(A')$. If a KEM adversary A runs in time t and makes at most q decryption oracle queries, then A is called a $KEM[t, q]$-adversary. We denote by $\mathbf{Adv}_{KEM}(A)$ the advantage of adversary A. We say that KEM is secure if this advantage is negligible for all efficient adversaries; that is, all $KEM[t, q]$-adversaries, with polynomially-bounded t, q. For more details about parameters and security definitions, see [5, 4].

Compared with the factoring based asymmetric cryptosystems, the elliptic-curve-based cryptosystems have the advantage of efficiency. PSEC-KEM is almost as efficient as any elliptic-curve-based asymmetric cryptosystem. Table 1 shows a comparison of PSEC-KEM with other elliptic-curve-based KEMs in terms of efficiency. We evaluate the efficiency of these schemes by counting the number of the scalar-multiplication over the underlying elliptic curve over the finite field.

Table 1: Comparison of Efficiency

Scheme	Encryption	Decryption
PSEC-KEM	2	2
ACE-KEM	5	3 (4*)
ECIES-KEM	2	1 (2*)

Note: In the case (∗), an additional scaler-multiplication in the underlying elliptic curve group is required to check whether a ciphertext is in the subgroup generated by the base point.

Table 2 shows that PSEC-KEM has been proven secure, assuming the weakest "number-theoretic" assumption, EC-CDH, among the elliptic-curve-based KEMs, although ACE-KEM can hold its security in the standard model, instead of in the random oracle model.

Table 2: Comparison of Underlying Security

Scheme	Underlying assumption(s)	Model
PSEC-KEM	EC-CDH	Random oracle
ACE-KEM	EC-DDH and UOWHF	Standard
ECIES-KEM	EC-GDH	Random oracle

Note: EC-CDH, EC-DDH and EC-GDH denote the elliptic curve versions of computational Diffie-Hellman, decisional Diffie-Hellman and gap Diffie-Hellman assumptions, respectively. UOWHF denotes the assumption of the existence of a universally one-way hash function family.

References

[1] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. http://shoup.net/papers/, 2001 Dec.

[2] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes, Proc. of Crypto'99, Springer-Verlag, LNCS 1666, pp. 535–554 (1999).

[3] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (v.2.1), ISO/IEC JTC1/SC27, N2563, http://shoup.net/papers/, 2001 Dec.

[4] ISO/IEC 18033-2 Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric ciphers.

[5] Specification of PSEC-KEM, submission to IEEE P1363a, Aug 2006.