Specification of EPOC: Efficient Probabilistic Public-Key Encryption

1 Introduction

This document describes a novel public-key encryption scheme, EPOC (Efficient Probabilistic Public-Key Encryption), which has three versions: EPOC-1, EPOC-2 and EPOC-3. EPOC-1 is a public-key encryption system that uses a one-way trapdoor function and a random function (hash function). EPOC-2 and EPOC-3 are public-key encryption systems that use a one-way trapdoor function, two random functions (hash functions) and a symmetric-key encryption (e.g., one-time padding and block-ciphers). We can use any secure hash function and symmetric-key encryption for EPOC. In our sample program and test data, we use an output-length variant hash function based on SHA-1 (see Section 10) for hash function, and the one-time-pad for symmetric-key encryption.

The encryption scheme described in this contribution is obtained by combining four results: one [32] is on the trapdoor function technique, and the others [16, 17, 30, 31] on conversion techniques using random functions.

2 Design Policy

One of the most important properties of public-key encryption is provable security. The strongest security notion in public-key encryption is that of non-malleability or semantical security against adaptive chosen-ciphertext attacks. Bellare, Desai, Pointcheval and Rogaway [4] show that semantical security against adaptive chosen-ciphertext attacks (IND-CCA2) is equivalent to (or sufficient for) the strongest security notion (NM-CCA2).

A promising way to construct a practical public-key encryption scheme semantically secure against adaptive chosen-ciphertext attacks (IND-CCA2) is to convert a primitive trap-door one-way function (such as RSA or ElGamal) by using *random functions*. Here, an ideally random function, the "random oracle", is assumed when proving the security, and the random function is replaced by a practical random-like function such as a one-way hash function (e.g., SHA-1 and MD5, etc.) when realizing it in practice. This approach was initiated by Bellare and Rogaway, and is called the *random oracle model* [5, 6].

Although security in the random oracle model cannot be guaranteed formally when a practical random-like function is used in place of the random oracle, this paradigm often yields much more efficient schemes than those in the *standard model* and gives an informal security guarantee of the schemes.

Two typical primitives of the trap-door one-way function are deterministic one-way permutation (e.g. RSA function) and probabilistic one-way function (e.g., ElGamal and Okamoto-Uchiyama functions).

Bellare and Rogaway presented a generic and efficient way to convert a trap-door one-way permutation to an IND-CCA2 secure scheme in the random oracle model. (The scheme created in this way from the RSA function is often called OAEP.) However, their method cannot be applied to probabilistic trap-door one-way functions such as ElGamal.

Recently the authors, Fujisaki and Okamoto [16, 17], and Okamoto and Pointcheval [30, 31] realized three generic and efficient measures to convert a probabilistic trap-door one-way function to an IND-CCA2 secure scheme in the random oracle model. One is conversion from a semantically secure (IND-CPA) trap-door one-way function to an IND-CCA2 secure scheme [16]. Another is from a trap-door one-way (OW-CPA) function and a symmetric-key encryption (including one-time padding) to an IND-CCA2 secure scheme [17]. The other is from a gap-trap-door one-way (OW-CPA) function and a symmetric-key encryption (including one-time padding) to an IND-CCA2 secure scheme [30, 31]. The latter two conversions can guarantee the total security of the public-key encryption system in combination with a symmetric-key encryption scheme.

EPOC has several outstanding properties as follows:

1. EPOC-1 is semantically secure or non-malleable against chosen ciphertext attacks (IND-CCA2 or NM-CCA2) in the random oracle model under the p-subgroup assumption, which is comparable to the quadratic residue and higher degree residue assumptions.

2. EPOC-2 with one-time padding (EPOC-2-OTP) is semantically secure or non-malleable against chosen ciphertext attacks (IND-CCA2 or NM-CCA2) in the random oracle model under the factoring assumption (or one-way assumption of the OU encryption function).

3. EPOC-2 with symmetric encryption (EPOC-2-SymE) is semantically secure or non-malleable against chosen ciphertext attacks (IND-CCA2 or NM-CCA2) in the random oracle model under the factoring assumption (or one-way assumption of the OU encryption function), if the underlying symmetric encryption is secure against passive attacks.

4. EPOC-3 with one-time padding (EPOC-3-OTP) is semantically secure or non-malleable against chosen ciphertext attacks (IND-CCA2 or NM-CCA2) in the random oracle model under the gap-factoring assumption (or gap-one-way assumption of the OU encryption function).

5. EPOC-3 with symmetric encryption (EPOC-3-SymE) is semantically secure or non-malleable against chosen ciphertext attacks (IND-CCA2 or NM-CCA2) in the random oracle model under the gap-factoring assumption (or gap-one-way assumption of the OU encryption function), if the underlying symmetric encryption is secure against passive attacks.

6. The trapdoor technique with EPOC is fundamentally different from any other previous scheme including RSA-Rabin and Diffie-Hellman-ElGamal.

7. If the parameters are appropriately chosen, the encryption and decryption speeds of EPOC are comparable to those of elliptic curve cryptosystems.

   Compared with OAEP (RSA) with small e (e.g.,$2^{32}+1$), although the encryption speed of EPOC is slower than that of OAEP, the decryption speed is faster than that of OAEP.

3 Notations

EPOC is specified by triplet $(\mathcal{G}, \mathcal{E}, \mathcal{D})$, where $\mathcal{G}$ is the key generation operation, $\mathcal{E}$ the encryption operation, and $\mathcal{D}$ the decryption operation.

We have three versions of EPOC: EPOC-1, EPOC-2 and EPOC-3. EPOC-1 is designed for key-distribution and EPOC-2 and EPOC-3 are designed for both usages: the combination of key-distribution and encrypted data transfer, as well as distribution of a longer key under limited public-key size.

In this specification, we use following notations.

- $a := b$: the value of b is substituted to a, or a is defined as b.
- $\mathbf{Z}$: the set of integers.
- $\mathbf{Z}/n\mathbf{Z} := \{0, 1, \ldots, n-1\}$.
- Let A, B be sets. $A \backslash B := \{x \mid x \in A \wedge x \notin B\}$.
- $(\mathbf{Z}/n\mathbf{Z})^* := \{1, 2, \ldots, n-1\} \backslash \{x \mid \gcd(x, n) \neq 1\}$.
- $\{0,1\}^*$ is the set of finite strings. $\{0,1\}^*$ is also denoted by $\mathbf{B}$.
- $\{0,1\}^i$ is the set of i bit length bit strings. $\{0,1\}^i$ is also denoted by $\mathbf{B}_i$.
- Let $a \in \mathbf{Z}$. $\mathbf{B}_i[a]$ denotes a bit string $(a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$ such that
$$a = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}$$
.
- Let $a := (a_{i-1}, a_{i-2}, \ldots, a_0) \in \mathbf{B}_i$. $\mathbf{I}[a]$ denotes an integer $b \in \mathbf{Z}$ such that
$$b = a_0 + 2a_1 + 2^2 a_2 + \cdots 2^{i-1} a_{i-1}$$
.
- If $a \in \mathbf{B}_i$, $|a| := i$.
- $a \equiv b \pmod{n}$ means $a - b$ is divided by n. $a := b \bmod n$ denotes $a \in \mathbf{Z}/n\mathbf{Z}$ and $a \equiv b \pmod{n}$.
- Let $a \in \mathbf{B}$ and $b \in \mathbf{B}$. $a||b$ denotes the concatenation of a and b. For example, $(0,1,0,0)||(1,1,0) = (0,1,0,0,1,1,0)$.
- Let $X \in \mathbf{B}$. $[X]^{mLen}$ denotes the most $mLen$ significant bits of X.
- Let $a \in \mathbf{B}_i$ and $b \in \mathbf{B}_i$. $a \oplus b$ means the bit-wise exclusive-or operation. (i.e., $a \oplus b \in \mathbf{B}_i$.)

4 Primitive Encryption Function

EPOC employs the OU (Okamoto-Uchiyama) encryption function as a primitive encryption function. The OU function is as follows:

4.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

[Input] Security parameter $k(= pLen) \in \mathbf{Z}$, which is a positive integer.

[Output] A pair of public-key, $(n, g, h, pLen, rLen) \in \mathbf{Z}^5$, and secret-key, $(p, g_p) \in \mathbf{Z}^2$.

The operation of $\mathcal{G}$, on input k, is as follows:

- Choose two primes p, q $(2^{k-1} \le p, q \le 2^k - 1)$, and compute $n := p^2 q$.
- Choose $g \in (\mathbf{Z}/n\mathbf{Z})^*$ randomly such that the order of $g_p := g^{p-1} \bmod p^2$ is p.
- Choose h_0 from $(\mathbf{Z}/n\mathbf{Z})^*$ randomly and independently from g. Compute $h := h_0^n \bmod n$.
- Set $pLen := k$.

Note: g_p is a supplementary parameter that improves the efficiency of decryption, since g_p can be calculated from p and g. h can be $g^n \bmod n$.

4.2 Encryption: $\mathcal{E}$

The input and output of $\mathcal{E}$ are as follows:

[Input] Plaintext $m \in \{0,1\}^{pLen-1}$ along with public-key $(n, g, h, pLen, rLen) \in \mathbf{Z}^5$.

[Output] Ciphertext $C \in \mathbf{Z}$.

The operation of $\mathcal{E}$, on input M and $(n, g, h, pLen, rLen)$ is as follows:

- Select $r \in \{0,1\}^{rLen}$ uniformly.
- Compute C:
$$C := g^{\mathbf{I}[m]} h^{\mathbf{I}[r]} \bmod n$$
.

4.3 Decryption: $\mathcal{D}$

The input and output of $\mathcal{D}$ are as follows:

[Input] Ciphertext $C \in \mathbf{Z}$ along with public-key $(n, g, h, pLen, rLen) \in \mathbf{Z}^5$ and secret-key $(p, g_p) \in \mathbf{Z}^2$.

[Output] Plaintext $m \in \{0,1\}^{pLen-1}$ or null string.

The operation of $\mathcal{D}$, on input C along with $(n, g, h, pLen, rLen)$, (p, g_p), and (p, g_p), is as follows:

- Compute $C_p := C^{p-1} \bmod p^2$, and $m' := \frac{L(C_p)}{L(g_p)} \bmod p$, where $L(x) := \frac{x-1}{p}$.
- Check whether the following equations hold or not:
$$m' \leq 2^{pLen} - 1.$$
- If it holds, output $\mathbf{B}_{pLen-1}[m']$ as decrypted plaintext. Otherwise, output null string.

5 Auxiliary Functions

In this section, we show auxiliary functions we use in this specification.

- (k bit) pseudo-random number generator.
- (k bit) prime number generator.
- Hash function.
- Symmetric encryption algorithm $SymE$.
- Primitive integer arithmetic algorithm.

6 Specification of EPOC-1

6.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

[Input] Security parameter $k(= pLen) \in \mathbf{Z}$, which is a positive integer.

[Output] A pair of public-key, $(n, g, h, HID, pLen, mLen, hLen, rLen) \in \mathbf{Z}^8$, and secret-key, $(p, g_p) \in \mathbf{Z}^2$.

The operation of $\mathcal{G}$, on input k, is as follows:

- Choose two primes p, q ($2^{k-1} \leq p, q \leq 2^k - 1$), and compute $n := p^2q$.
- Choose $g \in (\mathbf{Z}/n\mathbf{Z})^*$ randomly such that the order of $g_p := g^{p-1} \bmod p^2$ is p.
- Choose h_0 from $(\mathbf{Z}/n\mathbf{Z})^*$ randomly and independently from g. Compute $h := h_0^n \bmod n$.
- Set $pLen := k$. Set $mLen$ and $rLen$ such that $mLen + rLen \leq pLen - 1$.
- Select a (hash) function H: $\{0,1\}^{mLen+rLen} \longrightarrow \{0,1\}^{hLen}$, and HID is its identifier.

Note: g_p is a supplementary parameter that improves the efficiency of decryption, since g_p can be calculated from p and g. h can be $g^n \bmod n$. H can be fixed by the system and shared by many users.

6.2 Encryption: $\mathcal{E}$

The input and output of $\mathcal{E}$ are as follows:

[Input] Plaintext $M \in \{0,1\}^{mLen}$ along with public-key $(n, g, h, HID, pLen, mLen, hLen, rLen) \in \mathbf{Z}^8$.

[Output] Ciphertext $C \in \mathbf{Z}$.

The operation of $\mathcal{E}$, on input M and $(n, g, h, HID, pLen, mLen, hLen, rLen)$ is as follows:

- Select $R \in \{0,1\}^{rLen}$ uniformly, and compute $r := H(M||R)$.
- Compute C:
$$C := g^{\mathbf{I}[M||R]} h^{\mathbf{I}[r]} \bmod n$$
.

6.3 Decryption: $\mathcal{D}$

The input and output of $\mathcal{D}$ are as follows:

[Input] Ciphertext $C \in \mathbf{Z}$ along with public-key $(n, g, h, HID, pLen, mLen, hLen, rLen) \in \mathbf{Z}^8$ and secret-key $(p, g_p) \in \mathbf{Z}^2$.

[Output] Plaintext $M \in \{0,1\}^{mLen}$ or null string.

The operation of $\mathcal{D}$, on input C along with $(n, g, h, HID, pLen, mLen, hLen, rLen)$, (p, g_p), and (p, g_p), is as follows:

- Compute $C_p := C^{p-1} \bmod p^2$, and $X := \frac{L(C_p)}{L(g_p)} \bmod p$, where $L(x) := \frac{x-1}{p}$.
- Check whether the following equations hold or not:
$$X \leq 2^{mLen+rLen} - 1, \quad \text{and}$$
$$C = g^X h^{\mathbf{I}[H(X)]} \bmod n.$$
- If it holds, output $[\mathbf{B}_{mLen+rLen}[X]]^{mLen}$ as decrypted plaintext. Otherwise, output null string.

7 Specification of EPOC-2

7.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

[Input] Security parameter $k(= pLen) \in \mathbf{Z}$.

[Output] A pair of public-key, $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, rLen) \in \mathbf{Z}^{10}$ and secret-key, $(p, g_p) \in \mathbf{Z}^2$.

The operation of $\mathcal{G}$, on input k, is as follows:

- Choose two primes p, q ($2^{k-1} \leq p, q \leq 2^k - 1$) and compute $n = p^2 q$.
- Choose $g \in (\mathbf{Z}/n\mathbf{Z})^*$ randomly such that the order of $g_p := g^{p-1} \bmod p^2$ is p.
- Choose h_0 from $(\mathbf{Z}/n\mathbf{Z})^*$ randomly and independently from g. Compute $h := h_0^n \bmod n$.
- Set $pLen := k$. Set $rLen$ such that $rLen \leq pLen - 1$.
- Select (hash) functions H: $\{0,1\}^{mLen+rLen} \longrightarrow \{0,1\}^{hLen}$, and G: $\{0,1\}^{rLen} \longrightarrow \{0,1\}^{gLen}$, and the identifiers of H and G are HID and GID respectively.
- Let $SymE = (SymEnc, SymDec)$ be a pair of symmetric-key encryption and decryption algorithms with symmetric-key K, where the length of K is $gLen$. The identifier of $SymE$ is SEID. Let SEID $= 1$ denotes that $SymE$ is the one-time-pad.

  Encryption algorithm $SymEnc$ takes key K and plaintext X, and returns ciphertext $SymEnc(K, X)$. Decryption algorithm $SymDec$ takes key K and ciphertext Y, and returns plaintext $SymDec(K, Y)$. Here we assume that for any key K, function $SymEnc(K, \cdot)$ is one-to-one and onto.

Note: g_p is a supplementary parameter that improves the efficiency of decryption, since g_p can be calculated from p and g. h can be $g^n \bmod n$. H and G can be fixed by the system and shared by many users.

7.2 Encryption: $\mathcal{E}$

The input and output of $\mathcal{E}$ are as follows:

[Input] Plaintext $M \in \{0,1\}^{mLen}$ along with public-key $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, rLen) \in \mathbf{Z}^{10}$.

[Output] Ciphertext $C = (C_1, C_2) \in \mathbf{Z} \times \{0,1\}^{mLen}$.

The operation of $\mathcal{E}$, on input M, $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, rLen)$ is as follows:

- Select $R \in \{0,1\}^{rLen}$ uniformly, and compute $G(R)$ and $r := H(M||R)$.
- Compute $H(M||R)$.
- $$C_1 := g^{\mathbf{I}[R]} h^{\mathbf{I}[r]} \bmod n,$$
  $$C_2 := SymEnc(G(R), M)$$

Remark: A typical way to realize $SymE$ is one-time padding.
That is, $SymEnc(K, X) := K \oplus X$, and $SymDec(K, Y) := K \oplus Y$, where $\oplus$ denotes the bit-wise exclusive-or operation.

When $mLen$ is longer than $gLen$, we use an appropriate symmetric encryption (block cipher or stream cipher) rather than one-time padding.

7.3 Decryption: $\mathcal{D}$

The input and output of $\mathcal{D}$ are as follows:

[Input] Ciphertext $C = (C_1, C_2) \in \mathbf{Z} \times \{0,1\}^{mLen}$ along with public-key $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, rLen) \in \mathbf{Z}^{10}$ secret-key $(p, g_p) \in \mathbf{Z}^2$.

[Output] Plaintext $M \in \{0,1\}^{mLen}$ or null string.

The operation of $\mathcal{D}$, on input $C = (C_1, C_2)$ along with $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, rLen)$, and (p, g_p) is as follows:

- Compute $C_p := {C_1}^{p-1} \bmod p^2$, and $R' := \frac{L(C_p)}{L(g_p)} \bmod p$, where $L(x) := \frac{x-1}{p}$.
- Compute $M' := SymDec(G(R'), C_2)$.
- Check whether the following equations hold or not:

$$R' \le 2^{rLen} - 1,$$

$$C_1 = g^{R'} h^{\mathbf{I}[r']} \bmod n,$$

  where $r' := H(M' || \mathbf{B}_{rLen}[R'])$.
- If they hold, output M' as decrypted plaintext. Otherwise, output null string.

8 Specification of EPOC-3

8.1 Key Generation: $\mathcal{G}$

The input and output of $\mathcal{G}$ are as follows:

[Input] Security parameter $k (= pLen) \in \mathbf{Z}$.

[Output] A pair of public-key, $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, RLen, rLen) \in \mathbf{Z}^{11}$ and secret-key, $(p, g_p) \in \mathbf{Z}^2$.

The operation of $\mathcal{G}$, on input k, is as follows:

- Choose two primes p, q $(2^{k-1} \le p, q \le 2^k - 1)$ and compute $n = p^2 q$.
- Choose $g \in (\mathbf{Z}/n\mathbf{Z})^*$ randomly such that the order of $g_p := g^{p-1} \bmod p^2$ is p.
- Choose h_0 from $(\mathbf{Z}/n\mathbf{Z})^*$ randomly and independently from g. Compute $h := h_0^n \bmod n$.
- Set $pLen := k$. Set $RLen$ such that $RLen \le pLen - 1$.
- Select (hash) functions H: $\{0,1\}^{3k+kLen+RLen+mLen} \longrightarrow \{0,1\}^{hLen}$, and G: $\{0,1\}^{RLen} \longrightarrow \{0,1\}^{gLen}$ and the identifiers of H and G are HID and GID respectively.

- Let $SymE = (SymEnc, SymDec)$ be a pair of symmetric-key encryption and decryption algorithms with symmetric-key K, where the length of K is $gLen$. The identifier of $SymE$ is SEID. Let SEID = 1 denotes that $SymE$ is the one-time-pad.

  Encryption algorithm $SymEnc$ takes key K and plaintext X, and returns ciphertext $SymEnc(K, X)$. Decryption algorithm $SymDec$ takes key K and ciphertext Y, and returns plaintext $SymDec(K, Y)$. Here we assume that for any key K, function $SymEnc(K, \cdot)$ is one-to-one and onto.

Note: g_p is a supplementary parameter that improves the efficiency of decryption, since g_p can be calculated from p and g. h can be $g^n \bmod n$. H and G can be fixed by the system and shared by many users.

8.2 Encryption: $\mathcal{E}$

The input and output of $\mathcal{E}$ are as follows:

[Input] Plaintext $M \in \{0,1\}^{mLen}$ along with public-key $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, RLen, rLen) \in \mathbf{Z}^{11}$.

[Output] Ciphertext $C = (C_1, C_2, C_3) \in \mathbf{Z} \times \{0,1\}^{mLen} \times \{0,1\}^{hLen}$.

The operation of $\mathcal{E}$, on input M, $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, RLen, rLen)$ is as follows:

- Select $r \in \{0,1\}^{rLen}$ and $R \in \{0,1\}^{RLen}$ uniformly, and compute $G(R)$.
- Compute

$$C_1 := g^{\mathbf{I}[R]} h^{\mathbf{I}[r]} \bmod n,$$

$$C_2 := SymEnc(G(R), M),$$

$$C_3 := H(\mathbf{B}_{3 \cdot pLen}[C_1] || C_2 || R || M).$$

Remark: A typical way to realize $SymE$ is one-time padding.
That is, $SymEnc(K, X) := K \oplus X$, and $SymDec(K, Y) := K \oplus Y$, where $\oplus$ denotes the bit-wise exclusive-or operation.

When $mLen$ is longer than $gLen$, we use an appropriate symmetric encryption (block cipher or stream cipher) rather than one-time padding.

8.3 Decryption: $\mathcal{D}$

The input and output of $\mathcal{D}$ are as follows:

[Input] Ciphertext $C = (C_1, C_2, C_3) \in \mathbf{Z} \times \{0,1\}^{mLen} \times \{0,1\}^{hLen}$ along with public-key $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, RLen, rLen) \in \mathbf{Z}^{11}$ secret-key $(p, g_p) \in \mathbf{Z}^2$.

[Output] Plaintext $M \in \{0,1\}^{mLen}$ or null string.

The operation of $\mathcal{D}$, on input $C = (C_1, C_2)$ along with $(n, g, h, HID, GID, SEID, pLen, hLen, gLen, RLen, rLen)$ and (p, g_p) is as follows:

- Compute $C_p := {C_1}^{p-1} \bmod p^2$, and $R' := \frac{L(C_p)}{L(g_p)} \bmod p$, where $L(x) := \frac{x-1}{p}$.
- Compute $M' := SymDec(G(R'), C_2)$.
- Check whether the following equations hold or not:

$$R' \le 2^{rLen} - 1,$$

$$C_3 = H(\mathbf{B}_{3 \cdot pLen}[C_1] || C_2 || \mathbf{B}_{rLen}[R'] || M').$$

- If they hold, output M' as decrypted plaintext. Otherwise, output null string.

8.4 Session Like Method for EPOC-3 Encryption

We can use EPOC-3 as following session like method.

- Sender chooses uniform randomly $r \in \{0,1\}^{rLen}$ and $R \in \{0,1\}^{RLen}$.
- Sender computes $C_1 := g^{\mathbf{I}[R]} h^{\mathbf{I}[r]} \bmod n$ and $K := G(R)$, and sends C_1.
- Receiver decrypts R from C_1, and computes $K := G(R)$. [key sending phase finished]
- For each plaintext $M_i \quad (i = 1, 2, \ldots,)$, Sender computes $C_{2,i} := SymEnc(K, M_i)$ and $C_{3,i} := H(\mathbf{B}_{3 \cdot pLen}[C_1] || C_{2,i} || R || M_i)$, and sends $(C_{2,i}, C_{3,i})$.
- Receiver decrypts M_i by using K, and checks $C_{3,i} = H(\mathbf{B}_{3 \cdot pLen}[C_1] || C_{2,i} || R || M_i)$. [cipher communication phase finished]

9 Recommended Parameters

For EPOC-1/2/3, the security parameter, k, should be greater than 320 (i.e., the size of n should be greater than 960), and $hLen$ should be at least 128.

Here we will show two typical cases of parameters employed in our self-evaluation document: Type-A and Type-B of security parameters for EPOC-1/2/3. EPOC with Type-A parameters are provably secure in the strongest sense (IND-CCA2) under "weaker" assumptions, while EPOC with Type-B parameters are provably secure in the strongest sense (IND-CCA2) under "stronger" assumptions. EPOC with Type-B parameters enjoy better performance than EPOC with Type-A parameters.

The length of modulus n for the both types is 1152 bits.

- **[Type A parameters]** For EPOC-1, the message length ($mLen$) is 128 bits, random string length ($rLen$) is 80 bits and the hashed value length of h ($hLen$) is 832 bits.

  For EPOC-2 with one-time pad, $rLen = 128$ and the hashed value lengths of h and g ($hLen$ and $gLen$) are 832 and 128, respectively.

As for EPOC-3 with one-time pad, the random string lengths ($rLen$ and $RLen$) are 832 and 128 bits respectively, and the hashed value lengths of h and g ($gLen$ and $hLen$) are 128.

- **[Type B parameters]** For EPOC-1, the message length ($mLen$) is 128 bits, random string length ($rLen$) is 80 bits and the hashed value length of h ($hLen$) is 208 bits. For EPOC-2 with one-time padding (OTP), $rLen = 128$ and the hashed value lengths of h and g ($hLen$ and $gLen$) are 128. As for EPOC-3 with one-time padding (OTP), the random string lengths ($rLen$ and $RLen$) are 128 bits, and the hashed value lengths of h and g ($gLen$ and $hLen$) are 128.

10 Hash Function

We can use any random-like one-way functions H and G for EPOC. (As mentioned in subsection 2, EPOC can be proven to be secure if H and G are ideal random functions, while no formal security is guaranteed if they are practical random-like one-way functions.) In this subsection we will show a typical construction of function H with $hLen > 160$ out of SHA (NIST Secure Hash Algorithm), which was suggested by Bellare and Rogaway [6].

We denote by $\mathrm{SHA}_\sigma(x)$ the 160-bit result of SHA applied to x, except that the 160-bit "starting value" in the algorithm description is taken to be $ABCDE = \sigma$. Let $\mathrm{SHA}^l_\sigma(x)$ denote the first l-bits of $\mathrm{SHA}_\sigma(x)$. Fix the notation $< i >$ for i encoded as a binary 32-bit word. We define the function H as:

$$H(x) := \mathrm{SHA}^{80}_\sigma(< 0 > ||x)||\mathrm{SHA}^{80}_\sigma(< 1 > ||x)|| \cdots ||\mathrm{SHA}^{L_l}_\sigma(< l > ||x),$$

where $l = \lfloor \frac{3k}{80} \rfloor$, and $L_l = hLen - 80l$.

References

[1] Abdalla, M., Bellare, M. and Rogaway, P.: DHES: An Encryption Scheme Based on the Diffie-Hellman Problem, Submission to IEEE P1363a (1998, August)

[2] Adleman, L.M. and McCurley, K.S.: Open Problems in Number Theoretic Complexity,II (open problems: C7, O7a and O7b), Proc. of ANTS-I, LNCS 877, Springer-Verlag, pp.291-322 (1995).

[3] Ajtai, M. and Dwork, C.: A Public-Key Cryptosystem with Worst-Case/Average-Case Equivalence, Proc. of STOC'97, pp. 284-293 (1997).

[4] Bellare, M., Desai, A., Pointcheval, D., and Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes, Proc. of Crypto'98, LNCS 1462, Springer-Verlag, pp. 26–45 (1998).

[5] Bellare, M. and Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, Proc. of the First ACM Conference on Computer and Communications Security, pp.62–73 (1993).

[6] Bellare, M. and Rogaway, P. : Optimal Asymmetric Encryption, Proc. of Eurocrypt'94, LNCS 950, Springer-Verlag pp.92-111 (1995).

[7] Boneh, D., Durfee, G. and Howgrave-Graham, N.: Factoring $N = p^r q$ for Large r, Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.326-337 (1999)

[8] Chao, J., Matsuda, N. and Tsujii, S.: Efficient construction of secure hyperelliptic discrete logarithm problems, Proc. of ICICS'97, LNCS 1334, Springer-Verlag, pp.292-301 (1997).

[9] Chor, B. and Rivest, R.L.: A knapsack type public key cryptosystem based on arithmetic in finite fields, Proc. of Crypto'84, LNCS 196, Springer-Verlag, pp.54-65 (1985).

[10] Cohen, J. and Fischer.: A Robust and Verifiable Cryptographically Secure Election Scheme, FOCS, pp.372-382 (1985).

[11] Cryptography Using Compaq MultiPrime Technology in a Parallel Processing Environment, Enterprise Security Solutions, Electronic Commerce Technical Brief, Compaq Computer Corporation, http://www6.compaq.com/solutions/security/ (2000)

[12] Dolev, D., Dwork, C. and Naor, M.: Non-Malleable Cryptography, Proc. of STOC, pp.542–552 (1991).

[13] Demytko, N.: A New Elliptic Curve Based Analogue of RSA, Proc. of Eurocrypt'93, LNCS 765, Springer-Verlag, pp.40-49 (1994).

[14] Diffie, W. and Hellman, M.: New Directions in Cryptography, IEEE Trans. on Information Theory, IT-22, 6, pp.644–654 (1976).

[15] ElGamal, T.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Trans. on Information Theory, IT-31, 4, pp.469–472 (1985).

[16] Fujisaki, E. and Okamoto, T.: How to Enhance the Security of Public-Key Encryption at Minimum Cost, Proc. of PKC'99, LNCS 1560, Springer-Verlag, pp.53–68 (1999).

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

[18] Goldwasser, S. and Micali, S.: Probabilistic Encryption, JCSS, 28, 2, pp.270-299 (1984).

[19] IEEE P1363 Draft (D9), http://grouper.ieee.org/groups/1363/P1363/draft.html (1999).

[20] Joye, M., Quisquater, J.J., and Yung, M.: On the Power of Misbehaving Adversaries and Security Analysis of EPOC, Manuscript (February 2000).

[21] Koblitz, N.: Elliptic Curve Cryptosystems, Math. Comp., 48, 177, pp.203–209 (1987).

[22] Koyama, K. , Maurer, U. M. , Okamoto, T. and Vanstone, S. A.,: New Public-key Schemes based on Elliptic Curves over the Ring Zn, Proc. of Crypto'91, LNCS 576, Springer-Verlag, pp.252-266 (1992).

[23] Kurosawa, K., Ito, T. and Takeuchi, M.: Public Key Cryptosystem using a Reciprocal Number with the same Intractability as Factoring a Large Number, Cryptologia, 12, 4, pp.225-233 (1988).

[24] Loxton, J.H., Khoo, D.S.P., Bird, G.J. and Seberry, J.: A Cubic RSA Code Equivalent to Factorization, Journal of Cryptology, 5, 2, pp.139-150 (1992).

[25] Matsumoto, T. and Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption, Proc. of Eurocrypt'88, LNCS 330, Springer-Verlag, pp.419-453 (1988).

[26] McEliece, R.J.: A Public-Key Cryptosystem Based on Algebraic Coding Theory, DSN progress report 42-44, Jet Propulsion Laboratories, Pasadena (1978).

[27] Merkle, R.C. and Hellman, M.E.: Hiding Information and Signatures in Trapdoor Knapsacks, IEEE Trans. on Inform. Theory, 24, pp.525-530 (1978).

[28] Miller, V.S.: Use of Elliptic Curves in Cryptography, Proc. of Crypto'85, LNCS 218, Springer-Verlag, pp.417-426 (1985).

[29] Naccache, D. and Stern, J.: A New Public-Key Cryptosystem, Proc. of Eurocrypt'97, LNCS 1233, Springer-Verlag, pp.27-436 (1997).

[30] Okamoto, T. and Pointcheval, D.: BEST: A Generic Coversion to Achieve Chosen-Ciphertext Security, manuscript (2000).

[31] Okamoto, T. and Pointcheval, D.: The Gap Problems: A New Class of Problems for the Security of Cryptographic Schemes, manuscript (2000).

[32] Okamoto, T. and Uchiyama, S.: A New Public-Key Cryptosystem as Secure as Factoring, Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 308–318(1998).

[33] Patarin, J. and Goubin, L.: Trapdoor one-way permutations and multivariate polynomials, Proc. of ICICS'97, LNCS 1334, Springer-Verlag, pp.356-368 (1997).

[34] Patarin, J. and Goubin, L.: Asymmetric cryptography with S-Boxes, Proc. of ICICS'97, LNCS 1334, Springer-Verlag, pp.369-380 (1997).

[35] Peralta, R.: Bleichenbacher's improvement for factoring numbers of the form $N = PQ^2$ (private communication) (1997).

[36] Peralta, R. and Okamoto, E.: Faster Factoring of Integers of a Special Form, IEICE Trans. Fundamentals, E79-A, 4, pp.489-493 (1996).

[37] Pollard, J.L.: Manuscript (1997).

[38] Rabin, M.O.: Digital Signatures and Public-Key Encryptions as Intractable as Factorization, MIT, Technical Report, MIT/LCS/TR-212 (1979).

[39] Rivest, R., Shamir, A. and Adleman,L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Communications of the ACM, Vol.21, No.2, pp.120-126 (1978).

[40] R. D. Silverman: A Cost-Based Security Analysis of Symmetric and Asymmetric Key Lengths", RSA Laboratories, CryptoBytes, Bulletins, Number 13 (April 2000), http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html/.

[41] Smith, P. and Lennon, M.: LUC: A New Public Key System, Proc. of IFIP/SEC'93, pp. 103-117, North-Holland (1993).

[42] Williams, H.C.: A Modification of the RSA Public Key Encryption Procedure, IEEE Trans. on Inform. Theory, IT-26, 6, pp.726-729 (1980).

[43] Williams, H.C.: Some Public-Key Crypto-Functions as Intractable as Factorization, Proc. of Crypto'84, LNCS 196, Springer-Verlag, pp.66-70 (1985).