Initial report on the EPOC asymmetric encryption scheme*†

Juliette White

NES/DOC/RHU/WP3/011/2

1 Description

EPOC (Efficient Probabilistic Public-Key Encryption) in fact consists of three separate probabilistic asymmetric encryption schemes, namely EPOC-1, EPOC-2, and EPOC-3. The schemes are submitted by NTT Labs, Nippon Telegraph and ENS.

EPOC-1 uses a one-way trapdoor function together with a hash function, and is intended solely for key distribution. EPOC-2 uses a symmetric encryption scheme and uses the idea of EPOC-1 to securely distribute the key used for symmetric encryption of the message. It also uses a second hash function. EPOC-3 is a refinement of EPOC-2. The relationship between EPOC-1, EPOC-2, and EPOC-3 is very similar to the relationship between PSEC-1, PSEC-2 and PSEC-3, three asymmetric encryption schemes based on elliptic curves, submitted by the same authors.

The security of each of the three schemes depends on the assumption that a different problem is intractable. The authors do not detail explicitly the relative advantages and disadvantages of the three schemes.

The authors do not specify the hash functions or symmetric encryption scheme to be used, nor do they make any specifications concerning random number generation.

Preliminaries

Let \( \Gamma \) be the Sylow \( p \)-subgroup of \( (\mathbb{Z}/p^2\mathbb{Z})^* \). The set \( \Gamma \) then consists of those elements \( x \) in \( (\mathbb{Z}/p^2\mathbb{Z})^* \) such that \( x \equiv 1 \mod p \). Define the function \( L \) by

\[
L : \Gamma \rightarrow \mathbb{Z}/p\mathbb{Z} \\
x \mapsto \frac{x - 1}{p}
\]

This is an isomorphism between \( \Gamma \) and \( \mathbb{Z}/p\mathbb{Z} \).

We shall need the following two facts.

---

*The work described in this paper has been supported by the Commission of the European Communities through the IST program under contract IST-1999-12324

†The information in this document is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.
• Let $x \in \Gamma$ and $L(x) \neq 0 \mod p$. If 
\[ y = x^m \mod p^2 \text{ where } m \in \mathbb{Z}/p\mathbb{Z} \]
then 
\[ m = \frac{L(x)}{L(y)} \]

• If $g$ is a primitive root $\mod p^2$ and 
\[ g_p = g^{p-1} \mod p^2 \]
then there exists $r \in (\mathbb{Z}/p\mathbb{Z})^*$ such that $L(g_p) = r$.

1.1 EPOC-1

We have some public parameters $(H, pLen, mLlen, hLen, rLen)$, the public key is $(n, g, h)$ and the private key is $(p, g_p)$ where

• $p$ and $q$ are primes of length $pLen$
• $n = p^2q$
• $g$ is chosen at random from $\mathbb{Z}_n^*$ such that $g_p = g^{p-1} \mod p^2$ has order $p$ modulo $p^2$
• $h \equiv h_0^n \mod n$ where $h_0$ is chosen at random from $\mathbb{Z}_n^*$
• $pLen, mLlen, hLen$ and $rLen$ are length parameters such that $mLen + rLen \leq pLen - 1$
• $H$ is a hash function $H : \{0,1\}^{mLen+rLen} \rightarrow \{0,1\}^{hLen}$

Encryption

A message $M$ of length $mLen$ is encrypted as follows:

1. Choose random binary string $R$ of length $rLen$.
2. Compute ciphertext $C \equiv g^{M||R}h^{H(M||R)} \mod n$

Decryption

The decryption of a ciphertext $C$ to retrieve the message $M$ is as follows:

1. Compute $C_p \equiv C^{p-1} \mod p^2$
2. Compute $X \equiv \frac{L(C_p)}{L(g_p)} \mod p$.
3. Check whether or not $X < 2^{mLen+rLen}$ and $C \equiv g^Xh^{H(X)} \mod n$
4. If the above checks are OK then $M$ is equal to the $mLen$ most significant bits of $X$. 

2
Why decryption works

We shall ignore the hash function and random $R$ (it is easy to see that the explanation below still works when we include these). So we may assume $C \equiv g^M h^M \mod n$.

We also have $(h^{p-1})^M \equiv 1 \mod p^2$

since $h \equiv h_0^{p^2} \mod p^2$ and $h_0^{p(p-1)} \equiv 1 \mod p^2$.

Therefore $C_p \equiv (g_p)^M \mod p^2$.

But $L(g_p) \neq 0 \mod p$, so it follows that $M \equiv \frac{L(C_p)}{L(g_p)}$.

1.2 EPOC-2

For EPOC-2, the public key is the same as that for EPOC-1 together with an integer $gLen$, a hashing function $G \colon \{0, 1\}^{rLen} \to \{0, 1\}^{sLen}$ and a symmetric encryption/decryption scheme $S$ with key $K$ of length $gLen$. The private key is as for EPOC-1. (The submitters forget to include $mLen$ in the public key here in their documentation). The only other difference in the keys is that here, instead of the inequality concerning the parameter lengths for EPOC-1, we insist that $rLen \leq pLen + 1$.

Encryption

The encryption of a message $M$ is carried out as follows:

1. Choose a random string $R$ of length $rLen$.
2. Let $C_1 = g^R h^{H(M || R)}$.
3. Compute $C_2$, the encryption of $M$ under $S$ using the secret key $G(R)$.
4. Output ciphertext $C = (C_1, C_2)$

Decryption

The decryption of the ciphertext $C = (C_1, C_2)$ to retrieve the message $M$ is as follows:

1. Compute $C_p \equiv C_1^{p-1} \mod p^2$
2. Compute $R' \equiv \frac{L(C_p)}{L(g_p)} \mod p$
3. Decrypt $C_2$ using the symmetric encryption scheme $S$ with key $G(R')$ to get $M'$
4. Compute $r' = H(M' \parallel (rLen \text{ most sig bits of } R'))$
5. Check that
\[ R' \leq 2^{r\text{Len}} - 1 \]
\[ C_1 = g^{R'} h' \]

6. If these do hold, then output \( M = M' \)

It is easy to adjust the explanation for EPOC-1 to show why decryption works for EPOC-2.

1.3 EPOC-3

The public and private keys required for EPOC-3 are exactly the same as those required for EPOC-2 except that there is an extra length parameter \( R\text{len} \) with \( R\text{Len} \leq p\text{Len} - 1 \), and the hash functions \( H \) and \( G \) take input strings of a different length. In EPOC-3 we have

\[ H : \{0, 1\}^{3p\text{Len} + R\text{Len} + 2m\text{Len}} \rightarrow \{0, 1\}^{h\text{Len}} \]

and

\[ G : \{0, 1\}^{r\text{Len}} \rightarrow \{0, 1\}^{g\text{Len}} \]

Encryption

Encryption of message \( M \) is as follows:

1. Choose random binary strings \( r \) and \( R \) of lengths \( r\text{Len} \) and \( R\text{Len} \) respectively, and compute \( G(R) \)

2. Compute \( C_1 \equiv g^{R} h^r \mod n \)

3. Compute the symmetric encryption, \( C_2 \) of \( R \) using key \( G(R) \)

4. Compute \( C_3 = H((3p\text{Len most sig bits of})C_1 \parallel C_2 \parallel R \parallel M) \)

5. Output ciphertext \( C = (C_1, C_2, C_3) \)

Decryption

The decryption of \( C \) to retrieve the message \( M \) is as follows:

1. Compute \( C_p \equiv C_1^{p-1} \mod p^2 \)

2. Compute \( R' \equiv \frac{L(C_3)}{L(g^p)} \mod p \)

3. Decrypt \( C_2 \) using the symmetric encryption algorithm \( S \) with key \( G(R') \) to get \( M' \)

4. Check that
   
   \[ R' \leq 2^{r\text{Len}} - 1 \]
   \[ C_3 = H((3p\text{Len most sig bits of})C_1 \parallel C_2 \parallel (r\text{Len most sig bits of})R' \parallel M') \]

5. If these do hold then output message \( M = M' \)

Again we can easily adjust our earlier explanation to see why decryption works here.
2 Security

The three schemes have been proved in the random oracle model to be semantically secure against chosen ciphertext attack (the strongest notion of security for asymmetric encryption schemes) under, respectively, the $p$-subgroup assumption, the factoring assumption for $n = p^2q$, and the gap high-residuosity assumption. We also need to make certain assumptions concerning the symmetric encryption scheme and hash functions.

- **The $p$-subgroup assumption** The authors do not define this carefully. The only reference we have found indicates that it is a residuosity problem of some type in the Sylow $p$-subgroup of the multiplicative group of $\mathbb{Z}/p^2\mathbb{Z}$, but the precise type of residuosity problem is still unclear.

- **The factoring assumption** for $n = p^2q$ It is not known whether $n = p^2q$ is easier to factor than $n = pq$. The fastest algorithm is the Number Field Sieve method, the running time for which depends only on the size of $n$. The size of $n = p^2q$ can be the same as for $n = pq$

- **The gap high-residuosity assumption** The authors do not define this.

The problem of factoring $n = p^2q$ seems to have been well-studied whereas the other two problems do not appear to have been studied extensively. However it seems reasonable to assume that there is no easier way of solving these problems than factoring $n$.

Two types of parameters are provided for EPOC depending on the security level required, detailed in the submission. The value of $n$ recommended is 1024 bits. The cost of factoring a number of this size is claimed to be $2^{60}$ triple-DES encryptions.

For any of the three schemes, to find the secret key, even under active attacks it appears to be necessary to factor $n = p^2q$ and we assume that the problem of doing this is intractable. In order to retrieve a message text from a ciphertext it appears we need to solve a discrete log problem modulo $n$ which we also assume is an intractable problem. Even if the hash function $H$ is not collision-resistant, there is no clear way to produce two messages which give the same ciphertext. It is clear that the security of EPOC-2 and EPOC-3 depends on the security of the symmetric encryption scheme and on the security of the random number generator. Although it is possible to generate random numbers securely in polynomial time, such random number generators are slow, so their use may make the scheme impractical.

2.1 Summary

The EPOC schemes appear to be secure if the discrete log problem over $\mathbb{Z}_n$ is intractable and the hash functions, symmetric encryption schemes and random number generation are secure.