EPOC-2

Security Evaluation of the Encryption Scheme

Evaluator: Prof. Jean-Jacques Quisquater, Math RiZK, consulting

Scientific Support: Dr. François Koeune, K2Crypt

Executive summary.
This report is a full security evaluation of EPOC-2. The main conclusion is that our feeling about EPOC-2 is rather positive. It is however our belief that the system is a bit too young to be actually used in sensitive applications with a need of good level of confidentiality: we think it is really too early to use it for the encryption in the e-government in Japan. To be honest we like the algorithm but we firmly want to give a scientific and independent advise.

Contents

1 Preliminary Note and Introduction

In comparison with the signature scheme DSA, the encryption scheme EPOC presents several differences:

- EPOC is much more recent than DSA, with the consequence that the literature concerning it is much more limited. On the other hand, the fact that it has been proposed to several evaluation or standardization bodies (Nessie [1], IEEE P1363a [6], ...) implies that it has received a quite large attention from the scientific community since its birth. For example, its acceptance for the second round of Nessie is probably a witness of its quality.
- EPOC is in fact more a *family* of encryption schemes than a single scheme. As a matter of fact, several variants have been proposed to Nessie (EPOC-1, EPOC-2, EPOC-3 [5]), to IEEE P1363a [12](with some changes since first submission, ...). Even if these schemes belong undeniably to the same family, differences exist between them. The way parameters are handled (e.g. in the input to hash functions) also seems to have changed over time. In other words, we would be tempted to say that EPOC is still in an *unstable state*, in the sense that no two scientific papers on the subject discuss the same EPOC. This makes a general analysis somewhat difficult[1].
- EPOC leaves much more freedom in the choice of parameters or basic primitives (hash function, symmetric encryption primitive), as well as in the way these parameters will be generated (prime numbers, random values, ...). Specific choices seem however to be forced in standardized versions of EPOC.

2 Description of EPOC

Remark: As noted above, EPOC exists in different "flavors". The version we will discuss in this report is EPOC-2, as asked by the contract. [2]*, more precisely, the version of EPOC-2 submitted to the second round of Nessie [9].*

[1] In this perspective, it is interesting to note that Nessie has only accepted EPOC-2 for second round.

[2] Note that EPOC-1 and EPOC-3 are not supported any more by their submitters.

Asymmetric encryption schemes are usually much slower than symmetric ones. For this reason, they are rarely used solely to transmit large messages, but rather in combination with a symmetric scheme: a session key is first transmitted with the asymmetric scheme, and the full message is then encrypted with that session key and the symmetric scheme.

EPOC-2 is an hybrid encryption scheme, in the sense that it describes both the session key transmission and the subsequent encryption with symmetric scheme (EPOC-2 proposes Camellia as symmetric scheme, but other choices are possible as well). The security proof also covers this symmetric encryption, provided the underlying primitive satisfies given conditions (see below).

EPOC-2 is built starting from the Okamoto-Uchiyama [13] (denoted below as OU) scheme, which is provably one way under the factoring assumption, and applying to it techniques [11] to transform a one way asymmetric scheme and a semantically secure symmetric encryption function into an hybrid encryption scheme that is secure in the random oracle model against adaptive chosen-ciphertext attacks (these notions will be clarified in section 4).

2.1 EPOC-2 specification

A very detailed description of EPOC-2 (including bit-to-byte conversion, ...) can be found in [9]. For the sake of brevity and clarity, we will limit ourselves in this report to a "high-level" description of the scheme. We will however pay special attention to the parts which have been specifically changed (*tweaks*) between the first and second submission to Nessie, since we believe the impact of such changes on security must not be neglected. As a matter of fact, it is a well-known fact in cryptography that even the smallest change may introduce unexpected security weaknesses.

2.1.1 Parameters

EPOC-2 depends on a security parameter k, as well as two parameters, $hLen$ and $oLen$, corresponding to the sizes of mask and symmetric key. The size of these parameters will be discussed in section 3.

2.1.2 Key generation

1. Choose two primes $p, q \quad (2^{k-1} \leq p, q < 2^k, p \neq q)$.
2. Compute $n = p^2 q$.

3. Choose $g \in \mathbb{Z}_n^*$ such that the order of g_p in $\mathbb{Z}_{p^2}^*$ is p, where $g_p := g^{p-1} \bmod p^2$. (If $g_p \neq 1$, the order of g_p in $\mathbb{Z}_{p^2}^*$ is p).
4. Let $h := g^n \bmod n$.[3]
5. Set $pLen := k$.
6. Let $w := L(g_p)$, where $L(x) := \frac{x-1}{p}$.

The public key PK is the tuple $(n, g, h, pLen)$. The secret key SK is $(p, q, pLen, w)$.[4]

Remark: As an alternative, the second version of the specifications also allows to use a fixed value of g (provided the condition on the order of g_p is satisfied). In particular, the choice $g := 2$ is allowed.

2.1.3 Encryption

To encrypt a message M with public key PK, the following operations are performed:

1. Choose a random string $R \in \{0,1\}^{pLen}$.
2. Compute $C_2 := SymEnc(G(R), M)$, where $SymEnc$ is the symmetric encryption primitive, and G is a key-derivation function, built on the basis of a hash function (intuitively, it can be viewed as a hash function with variable-length output).
3. Compute $r := H(M||R||C_2)$,[5] where H is a mask generation function, built on the basis of a hash function (intuitively, it can be viewed as a hash function with variable-length output).
4. Compute $C_1 := g^R h^r \bmod n$.

The ciphertext is the pair (C_1, C_2).

[3] The first version of the specifications defined $h := h_0^n \bmod n$, where h_0 was chosen randomly and independently from g.

[4] This is a minor change since the first version, where the last member of the secret key was g_p rather than w.

[5] The string C_2 was initially (in first version of the specs.) not part of the input to H.

2.1.4 Decryption

To decrypt ciphertext (C_1, C_2) with secret key SK, the following operations are performed:

1. Check that $0 \leq C_1 < n$; otherwise, output null string and stop[6].
2. Compute $C_p := C_1^{p-1} \bmod p^2$.
3. Compute $R' := \frac{L(C_p)}{w} \bmod p$, where $L(x) := \frac{x-1}{p}$.
4. Check that $0 \leq R' < 2^{pLen-1}$; otherwise, output null string and stop.
5. Compute $M' := SymDec(G(R'), C_2)$, where $SymDec$ is the symmetric decryption primitive.
6. Check that $C_1 = g^{R'} h^{H(M'||R'||C_2)} \bmod q$; otherwise, output null string and stop[7].

The plaintext is given by M'.

2.1.5 Additional specifications

In addition, the second version of the specifications defines how to build H and G from a hash function, and recommends one hash function (SHA-1) and two possible symmetric encryption schemes: Camellia (another Nessie candidate, see [1]) and one-time-pad (OTP) [8].

2.1.6 Why does decryption work?

The fact that the decryption operation is the inverse of encryption results from the "logarithmic" character of the function L in the p-Sylow subgroup of $\mathbb{Z}_{p^2}^*$:

$$\Gamma := \{x \in \mathbb{Z}_{p^2}^* | x \equiv 1 \bmod p\}.$$

We refer the reader to [13] for more details.

[6] This test was initially not performed.

[7] In the first version of the specs, this test was performed with modulus n rather than q.

2.1.7 Comments on the changes between first and second version of specifications

This section briefly reviews the aforementioned changes between first and second submission to Nessie and comments on their potential impact on security.

Value of h fixed to $g^n \bmod n$: we note that this choice of h was explicitly allowed in the first version; this peculiar case should therefore have no impact on security.

Use of w rather than g_p in secret key : this minor change provides a performance improvement; as a matter of fact, it avoids the need to recompute $L(g_p)$ at every decryption; moreover, since w can be immediately deduced from g_p and p, this has no impact on security.

Use of C_2 as part of the input to H : since proofs in the random oracle model assume the hash functions to be perfect, this change has no impact on security proofs. An impact on *practical security* is however theoretically possible, although we do not see how this change could be exploited.

Integrity check performed with modulus q rather than n : we note that the security proof of [4] (discussed below), is performed with this definition of EPOC-2, including the integrity check with modulus q.

3 Parameters size

In [9], the following parameter sizes are recommended:

- $k = 384$;
- $hLen$ (i.e. bit length of the mask generation function output) $= 960$; if other values are chosen, it is recommended that $hLen \geq 2pLen + 32$;
- $oLen$ (i.e. bit length of the key of SymEnc, or, in other words, bit length of the key derivation function output) $= 128$.

Setting $k = 384$ (i.e. $|n| = 1152$) seems sufficient for nowadays' communications, in view of current state-of-the-art in factorization algorithms. It may however become insufficient in the future (within a few years), and we would recommend to leave the door open to larger values in a standard definition, as

well as to *use* larger values to encrypt documents whose long-term security has to be preserved.

Other parameters seem adequate. It seems however worth noting that these values have been increased since the first Nessie submission.

4 Security arguments

4.1 Security proof in the random oracle model

Ideally, we would like cryptographic schemes to be provably secure in the sense that it is possible to turn a breaking of the scheme into the solution of some problem believed to be hard (e.g. factorization of large integers). By contraposition, this would prove that, if the underlying problem is actually hard, then the scheme is secure.

Unfortunately, such security proofs are very difficult to obtain, and the corresponding cryptographic schemes are often hopelessly inefficient. As a compromise, it has become popular to prove security in an idealized model, in which hash functions are assumed to be perfect. This model is known as the random oracle model [2]. Similarly, the underlying symmetric encryption scheme is also assumed to be perfect, although in a less demanding sense (security against passive attacks).

As these perfect functions will in the real world be replaced by practical hash functions (i.e. SHA-1) and encryption schemes, such proofs become mathematically unsound in practical applications, and are therefore a purely heuristic argument. They are nonetheless usually considered as a good witness of security.

Levels of resistance An important point to settle in a security proof is a precise definition of what we mean by a breaking of the scheme. Several security levels have been defined in the literature, and the one we will be interested in with EPOC is that of *chosen-ciphertext security against an adaptative adversary*. The scenario we consider is the following: in a first stage, the adversary receives the public key and runs for some time, with the permission to ask to a decryption oracle to decrypt messages of his choice; the adversary then outputs two messages. One of these messages is chosen at random, encrypted with the public key and sent back to the adversary. The adversary's goal, in the second phase, is to discover which one of the messages was encrypted; to achieve this goal, he is also allowed in this phase to make requests to the decryption oracle (this is the

meaning of the term *adaptative adversary*), with the restriction that he is not allowed to submit the challenge ciphertext. The adversary is said to break the scheme if he is able to discover the good message with probability significantly greater than one-half. Note that this is a very strong security notion.
Remark: In the random oracle version of this security proof, the adversary has also access to random oracles modelizing the perfect hash functions.

Tightness of the reduction Another non-negligible question in such security proofs is the tightness of the reduction. Intuitively, we would like to know how much easier it is to break the scheme than to solve the underlying hard problem. One possible characterization is the following.

We define the advantage, ϵ, to be the difference of the probability that the adversary can guess correctly which of two messages encrypts to a given ciphertext from one half. The adversary is then said to (t, q_d, ϵ)-break the scheme if he is able, with running time t and q_d queries to the decryption oracle, to obtain advantage ϵ in solving the above challenge. A cryptographic scheme is said to be (t, q_d, ϵ)-secure if there exist no such adversary. Similarly, an algorithm is said to (t, ϵ)-solve a problem if it has running time t and advantage ϵ.

By comparing these factors, we can obtain a measurement of the tightness of the reduction.

4.2 EPOC's security proof

4.2.1 Underlying hard problem

EPOC-2's security relies on the intractability of factorizéing $n = p^2q$, for large primes p and q.

According to [14], *"The fastest known algorithm for factorizing large integers is the Number Field Sieve. The time taken to factor an integer n is given by $L_n[\frac{1}{3}, 1.526]$, which is subexponential. [...] The speed of the Number Field Sieve depends of the magnitude of n rather than that of the prime factors, and there is currently no faster method for factoring a number of the form $n = p^2q$ than one of the form $n = pq$. However if a faster method is found, the speed of which depends on the size of factors, then the key lengths have to be adjusted appropriately for those primitive relying on the difficulty of factoring of a number of the form $n = p^2q$.*

Remark: variants of EPOC (EPOC-1, EPOC-3, ...) rely on other hard problems (p-subgroup assumption, gap-factoring assumption, ...). We will however

not treat these cases in this report.

4.2.2 Security proof

EPOC-2's Nessie submission does not provide information on the tightness of the reduction, but refers to [11, 10] to obtain them. Unfortunately, these papers only provide information on the tightness of reducing EPOC-2 to the OU encryption scheme. More precisely, they show that, if OU is (t_1, ϵ_1)-secure (in the sense of one-wayness) under the factoring assumption, and if the symmetric encryption scheme is (t_2, ϵ_2)-secure (in the sense of "find-guess"), then EPOC-2 is $(t, q_g, q_h, q_d, \epsilon)$-secure against adaptative chosen-ciphertexts attacks, under the factoring assumption, where

$$t = \mathsf{min}(t_1, t_2) - O((q_g + q_h)(l_1 + l_2)),$$
$$\epsilon = (2(q_g + q_h)\epsilon_1 + \epsilon_2 + 1)(1 - 2\epsilon_1 - 2\epsilon_2 - 2^{-2k-l_2})^{-q_d} - 1,$$

and q_g and q_h are the number of hash queries to oracles G and H, q_d is the number of queries to the decryption oracle, and l_1 and l_2 are the respective sizes of the messages for the OU scheme and the symmetric encryption scheme ([14]).

The next step is therefore to check the tightness of reducing OU to the factorization of $n = p^2q$. However, moving back to the original OU paper [13], we only find "non-quantitative" proofs (we mean, proofs showing that breaking the scheme is equivalent to solving the hard problem, but without mentioning the tightness of the reduction). It is thus our belief that the initial Nessie submission did not allow to obtain a measure of the tightness of the reduction.

A very recent paper by Fujisaki ([4], Nov. 2001) filled this lack. The paper shows that, for a slight variant of EPOC-2 (a precise description of this variant is a bit too complex to fit in this report, but, in short, it consists in a restriction in the choice of g; taking $g = 2$ is one way of satisfying this restriction. Moreover, the underlying "perfect" symmetric scheme is built by applying a XOR with a stream derived from a "perfect" hash function.), theorem 4.1 holds. The authors also note that this is a much tighter reduction than what was first expected from the combination of reductions.

Theorem 4.1 *Suppose that there exists an adversary that $(t, q_g + q_h, q_d, \epsilon)$-breaks EPOC-2 (more exactly, the slight variant described above) in the random oracle model under the adaptative chosen-ciphertext attack scenario. Then there is a uniform algorithm for factoring n with running time t_B and advantage ϵ_B*

such that

$$\begin{aligned} t_B &= t(k) + (q_g + q_h)T_{gcd,n} + q_d q_h(T_{\epsilon,q} + T_{gcd,n}) \qquad \textit{and} \\ \epsilon_B &= \frac{\epsilon(k)}{3}\rho(1 - 2^{-3k+1})(1 - 2^{-\gamma})q_d, \end{aligned}$$

where

- $\rho \approx 1$
- γ *is the smallest integer such that*

  $$(\frac{1}{2})^{\gamma} \leq \frac{1}{o_q(g)} + (\frac{1}{2})^{2k+const}, \text{ and } o_q(g) \text{ is the order of } g \text{ in } \mathbb{Z}_n^*$$

- q_g *(resp.* q_h*) denotes the total number of requests to random oracles G (resp. H),*
- q_d *denotes the number of requests to the decryption oracle,*
- $T_{gcd,n}$ *denotes the time to compute the great common divisor between two integers of length* $|n|$*, and*
- $T_{\epsilon,q}$ *denotes the time to check the equation of the form*

  $$C_1 \equiv g^{\sigma} h^{H} \bmod q.$$

This proof has been reviewed by Dent [3], in the framework of the Nessie project, and was apparently approved[8]. For the sake of completeness, we would however like to point out that these documents [4, 3] are for the moment still internal to Nessie, and have therefore not yet received the "open review" of the scientific community.

4.2.3 Limitations of a security proof

Independently of the above remarks on the heuristic character of a proof in the random oracle model, it is really worth pointing out that a security proof does not guarantee that the system is absolutely immune to any type of attack. Weaknesses may appear, based on (often implicit) incorrect assumptions, insufficiently comprehensive model, ... A very good illustration of these limitations will be described in section 5.1, where an attack against a preliminary, but nevertheless proved secure, version of EPOC is described.

[8] An initial comment reported an error in the proof, but this comment was later withdrawn.

5 Overview of attacks

In the current literature, we are only aware of one attack against a preliminary version of EPOC. This attack is however a good illustration of the limitations of formal proofs of security.

5.1 Misbehaving adversaries attack

The weakness exploited by this attack [7] is a behavior of the adversary which was not taken into account in the design of the security proof: as a matter of fact, the security model did not assume an adversary could present an *invalid* ciphertext to the decryption oracle. As this case is out of the scope of the security model, it is not covered by the security proof. This allows, without highlighting a failure in the proof itself, to devise anyway a potentially damaging attack.

Description of the attack *Remark:* In the preliminary version of EPOC that we are attacking, the check $R' \leq 2^{rLen} - 1$ was not performed.

The encryption process (see section 2.1.3) assumes that the value R, from which the symmetric key is derived, is smaller than 2^{pLen-1}. What happens if a larger R is encrypted ?

Let $\hat{R}$ denote such a larger R, and let $\hat{C_1}(= g^R h^r \bmod n)$ denote the corresponding ciphertext part. The decryption of $\hat{C_1}$ yields the value

$$R = \frac{L(\hat{C_1}^{p-1} \bmod p^2)}{L(g_p)} \bmod p.$$

We have $R = \hat{R} \bmod p$. As $\hat{R} \geq p$, $R \neq \hat{R}$ and the test $\hat{C_1} \stackrel{?}{=} g^R h^{H(M||R)}$ will fail. The decryption algorithm will thus output the null string.

This can be exploited by an adversary as follows. Since the secret prime p is a $pLen$-bit number, he knows that p lies in the interval $I_0 =]2^{pLen-1}, 2^{pLen}[$. So the adversary chooses a value $\hat{R} \in I_0$ and performs an encryption with it. If the message can be decrypted then he knows that $\hat{R} < p$; otherwise he knows that $\hat{R} \geq p$. He then reiterates the process with $I_1 =]\hat{R}, 2^{pLen}[$ or $I_1 =]2^{pLen-1}, \hat{R}[$, respectively. And so on ... until the interval becomes small enough to guess the correct value of p. (For example, with a 1024-bit modulus n, at most 340 ciphertexts are necessary to recover the secret key.)

Countermeasures A simple way of preventing this attack (suggested by [7] and actually applied to subsequent versions of EPOC), is to explicitly check if $R' \leq 2^{rLen} - 1$ at decryption stage. In this way, all invalid values of $\hat{R}$ will be refused by the decryption oracle, no matter whether $\hat{R} < p$ or $\hat{R} \geq p$.

Concluding remark Although EPOC was finally modified to resist against this attack, this illustrates well the fact that even security proofs have limitations. As a matter of fact, they are only proofs in a given model, and under (sometimes implicit) specific assumptions. If the model is insufficiently comprehensive, attacks may appear that "turn around" the security proof by not satisfying its basis hypotheses.

6 Conclusion

Our feeling about EPOC-2 is rather positive, as seems confirmed by the positive comments it received from groups studying it (EC Nessie, IEEE P1363a). It is however our belief that the system is a bit too young to be actually used in sensitive applications with a need of good level of confidentiality: we think it is really too early to use it for the encryption in the e-government in Japan. To be honest we like the algorithm but we firmly want to give a scientific and independent advise.

It is usually good practice in cryptography to let some time (typically, a few years, at least two years without change in the fully published algorithm) elapse between the first appearance of a new cryptosystem and its actual use in practical applications. This leaves time for attacks to be published before they can do devastating damage. The scientific community today has enough publication channels for publishing such an attack, if any. Problems related with the implementations need also to be studied. See, for instance, the paper just presented at RSA 2002: `http://www.dice.ucl.ac.be/crypto/publications/2002/paradox2.pdf`

The fact that a security proof exists is certainly a non-negligible advantage to give confidence in the system's security. However, as was shown above, these proofs must not be considered as an absolute guarantee. Previous examples exist in the literature of provably-secure systems that were later the victim of unexpected attacks.

Another problem with EPOC is its unstable character: as mentioned above, several versions of EPOC co-exist, and even a given member of the family (e.g., EPOC-2) is still subject to minor modifications. We have for example described how such modifications (tweaks) have been introduced for the second round of Nessie. It is a well-known fact in cryptography that even the smallest modification may leave the door open to an unexpected attack.

Our advice would therefore be not to use the system yet, but at least wait for the outcome of the Nessie project (end 2002) and P1363a working group, and analyze their conclusions before actually using EPOC. Once a stable version of EPOC will be available, an independent study, carefully checking if results in the literature still apply to this version, should be performed.

References

[1] *Nessie (New European Schemes for Signatures, Integrity, and Encryption)*, Project funded by the European Community, see `http://www.cryptonessie.org/`.

[2] M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, Proc. of First Annual Conference on Computer and Commmunications Security, 1993, ACM.

[3] A. Dent, *An evaluation of EPOC-2*, Part of Nessie security evaluation (will appear at `http://www.cryptonessie.org/`), November 2001.

[4] E. Fujisaki, *Chosen-chiper security of EPOC-2*, Submitted as comment to Nessie, November 2001.

[5] E. Fujisaki, Kobayashi T., H. Morita, Oguro H., T. Okamoto, S. Okazaki, D. Pointcheval, and S. Uchiyama, *EPOC: Efficient probabilistic public-key encryption*, Submission to Nessie. Available at `https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions.html`, August 2000.

[6] IEEE, *IEEE P1363a: Standard specifications for public-key cryptography: Additional techniques*, See `http://grouper.ieee.org/groups/1363/P1363a/index.html`.

[7] M. Joye, J.-J. Quisquater, and M. Yung, *On the power of misbehaving adversaries and security analysis of the original EPOC*, Topics in Cryptology - CT-RSA 2001, April 2001, pp. 208–222.

[8] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of applied cryptography*, CRC Press, 1997.

[9] NTT Corporation NTT Information Sharing Platform Laboratories, *EPOC-2 specification*, Submission to the second round of Nessie, October 2001.

[10] T. Okamoto and E. Fujisaki, *How to enhance the security of public-key encryption at minimum cost*, Proc. of PKC '99 (Berlin), Springer-Verlag, 1999, Lecture Notes in Computer Science Volume 1560, pp. 53–68.

[11] ———, *Secure integration of asymmetric and symmetric encryption schemes*, Advances in Cryptology - Crypto '99 (Berlin), Springer-Verlag, 1999, Lecture Notes in Computer Science Volume 1666, pp. 535–554.

[12] T. Okamoto and S. Uchiyama, *EPOC: Efficient probabilistic public-key encryption*, Submission to IEEE P1363a. Latest draft available at `http://grouper.ieee.org/groups/1363/P1363a/draft.html`.

[13] ———, *A new public-key cryptosystem as secure as factoring*, Advances in Cryptology - Eurocrypt '98 (Berlin), Springer-Verlag, 1998, Lecture Notes in Computer Science Volume 1403, pp. 308–318.

[14] B. Preneel, B. Van Rompay, L. Granboulan, G. Martinet, S. Murphy, R. Shipsey, J. White, M. Dichtl, P. Serf, M. Schafheutle, E. Biham, O. Dunkelman, V. Furman, M. Ciet, J.-J. Quisquater, F. Sica, L. Knudsen, and H. Raddum, *Security evaluation of NESSIE first phase*, Nessie deliverable D13, September 2001.