Evaluation Report on the **EPOC Cryptosystem**

1 Introduction

This document is an evaluation of the **EPOC Cryptosystem**. Our work is based on the analysis of documents [11, 12], which provide both the specification and self-evaluation of the scheme, as well as on various research paper such as [28, 16, 17, 18, 26, 27], where additional security arguments can be found. The present report is organized as follows: firstly, we briefly review the cryptosystem; next we discuss the security level of the cryptographic primitive which underlies the scheme and analyze its relation to the difficulty of factoring; finally, we evaluate the security level of the scheme itself in the light of strong security notions such as semantic security and security against adaptive chosen-ciphertext attacks. This is as requested by IPA.

2 Brief description of the scheme

2.1 Specification review

EPOC is based on the hardness of various problems related to factoring integers n of the form $n = p^2q$, where p and q are prime numbers, approximately of the same size. It uses as a building block the Okamoto-Uchiyama public-key scheme which has appeared in [28]. To make things more precise, we need some notation from [11]: p and q are primes whose size in bits is k, n is p^2q, g is an element of $\mathbb{Z}_n^\star$, such that $g_p = g^{p-1} \bmod n$ is of order p modulo p^2 and h is an n-th power modulo n, built from a randomly chosen element h_0 of $\mathbb{Z}_n^\star$ by setting $h = h_0^n \bmod n$. The basic function f on which EPOC is based is defined by

$$\begin{array}{rcl} f : \{0,1\}^k \times \{0,1\}^\ell & \longrightarrow & \mathbb{Z}_n^\star \\ (x, r) & \longmapsto & g^x h^r \bmod n \end{array}$$

Thus, the public key of EPOC is a tuple (n, g, h, k, ℓ), where n, g, h, k are as above and ℓ is the bit-length of the second argument to f. The latter is usually $> k$ but it

is always assumed that $\frac{\ell}{k}$ is bounded by a small constant. There are additional items in the public key, consisting of hash function identifiers, which we ignore at this point. We will bring more precision on these elements at a later stage of our report. We have the following result from [28]:

Theorem 1 *Granted the factorization of n, one can recover x from $y = f(x, r)$ as $\frac{L(y^{p-1} \bmod n)}{L(g_p)}$, where $L(u)$ is $\frac{u-1}{p} \bmod p$.*

Proof: Note that $L(u)$ is defined on the subgroup of $\mathbb{Z}_n^\star$ consisting of integers u such that $u = 1 \bmod p$ and that $L(uv) = L(u) + L(v) \bmod p$. Now, $y^{p-1} = 1 \bmod p$ and similarly, $g_p = g^{p-1} = 1 \bmod p$. Thus both $L(x)$ and $L(g_p)$ can be computed. Furthermore,

$$L(y^{p-1}) = L(g^{(p-1)x} h^{p-1}) = L(g^{(p-1)x} h_0^{(p-1)n})$$

Since $h_0^{(p-1)n} = 1 \bmod p^2$, we get

$$L(y^{p-1}) = L(g_p^x) = x \cdot L(g_p) \bmod p$$

hence the result.

If the factorization of n remains unknown, computing x from $f(x, r)$ is believed to be hard.

Before going further, we introduce a more formal framework, that will be useful when we later perform the security analysis. A public-key encryption scheme on a message space $\mathcal{M}$ consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- the key generation algorithm $\mathcal{K}(1^k)$ outputs a random pair of secret-public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter k
- the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(M; R)$ outputs a ciphertext C corresponding to the plaintext $M \in \mathcal{M}$, using random coins R
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(C)$ outputs the plaintext M associated to the ciphertext C.

Thus, the key generation algorithm $\mathcal{K}(1^k)$ of the EPOC Cryptosystem produces, on input k, a public key pk consisting of an integer n of the form $p^2 q$, where p and q are k-bit primes, and of two elements g, h of $\mathbb{Z}_n^\star$, such that $g_p = g^{p-1} \bmod n$ is of order p modulo p^2 and h is a random n-th power modulo n. Further integers ℓ_R and ℓ_M indicate the respective bit-lengths of R and M, with $\ell_R + \ell_M \leq k - 1$. The secret key $\mathsf{sk} = (p, g_p)$ provides the factorization of n and the value of g_p. The latter is actually not needed, since it can be computed from p and public parameters. It is included here in order to make the decryption algorithm more efficient.

We now turn to encryption and decryption. At this point, we should mention that EPOC actually includes three different schemes.

- EPOC-1 is a public key encryption scheme
- EPOC-2 and EPOC-3 are hybrid encryption schemes

We first describe EPOC-1. EPOC-1 uses a hash function H, whose inputs are $\ell_R + \ell_M$-bit long and whose outputs are ℓ-bit long. Therefore, the public key includes a reference to the identifier of the hash function. Encryption $\mathcal{E}_{\mathsf{pk}}(M; R)$ computes C as:

$$g^{M||R} h^{H(M||R)} \bmod n = f(M||R, H(M||R))$$

Decryption recovers $M||R$, as explained in theorem 1, checks that it is of the appropriate bit-length $\ell_R + \ell_M \leq k - 1$ and verifies that C comes from M and R through the above formula.

EPOC-2 uses two hash functions H and G and a symmetric encryption scheme E. Accordingly, the public key includes a reference to the identifiers for the hash functions and encryption scheme and an indication of the length of the various inputs and outputs. Encryption $\mathcal{E}_{\mathsf{pk}}(M; R)$ computes C_1 as:

$$C_1 = g^R h^{H(M||R)} \bmod n = f(R, H(M||R))$$

From R, a key k for the symmetric encryption scheme E is derived as $\mathsf{k} = G(R)$. Using this key, a symmetric encryption step produces a cryptogram $C_2 = E_{\mathsf{k}}(M)$. The final ciphertext C is the pair (C_1, C_2).

Decryption $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering k. Granted the secret key p, one can recover R, as explained in theorem 1 and check that it is of the appropriate bit-length $\ell_R \leq k - 1$. From R, k is computed and, once k has been retrieved, one can obtain the plaintext M by applying the decryption algorithm of the symmetric encryption scheme. Before outputting the result, one verifies that C_1 comes from M and R through the above formula.

EPOC-3 is similar to EPOC-2 but produces a ciphertext C with three components $C = (C_1, C_2, C_3)$. Encryption uses an additional random element r, of bit-length ℓ: $\mathcal{E}_{\mathsf{pk}}(M; R, r)$ computes C_1 as

$$C_1 = g^R h^r \bmod n = f(R, r)$$

From R, a key k for the symmetric encryption scheme E is derived as $\mathsf{k} = G(R)$ and a cryptogram $C_2 = E_{\mathsf{k}}(M)$ is produced. Finally, a hash value $C_3 = H(C_1||C_2||R||M)$, of bit-length ℓ_H, is computed and the ciphertext C is the triple (C_1, C_2, C_3).

Decryption $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering k. Granted the secret key p, one can compute R, as explained in theorem 1 and check that it is of the appropriate bit-length $\ell_R \leq k - 1$. From R, k is computed and, once k has been retrieved, one can obtain the plaintext M by applying the decryption algorithm of the symmetric encryption scheme. Before outputting the result, one verifies that the hash value C_3 is correct.

At this point, we wish to note that we have not been too careful with notations in the above description. For example, writing the exponent of g as $M||R$ treats $M||R$ as a bit string or a byte string, whereas it is actually an integer. Document [11] provides the adequate conversion routines. We believe that our approach is suitable for performing a high level security analysis. This is why we use simplified notations and ignore type conversions.

2.2 Comments on the specification

Document [11] is clearly written and should actually ensure a minimal interoperability between different implementations. The description of key generation might be found a bit sketchy for implementors. For example, no additional constraint on $p-1$ is proposed, whereas one would expect that it is not a product of small primes. Similarly, the exact steps that generate parameter g are ignored. Additionally, there is a discrepancy between the submission under review and the research paper [28], which might have undesirable consequences in terms of security: in EPOC, parameter h is constructed from a randomly chosen $h_0 \in \mathbb{Z}_n^\star$ as $h = h_0^n \bmod n$ and "can be" $g^n \bmod n$, whereas this choice is mandatory in [28]. We will return to this feature further on.

Since there are three versions of EPOC, users might be a bit puzzled. No indication is given of the respective situations where one scheme would be more appropriate than the other. What can be observed, and clearly appears in the appendix of [12], is that EPOC-1 and EPOC-2 include reencrypting as a part of decryption, which is a rather undesirable feature in terms of efficiency. This is not a drawback in terms of security, unless the check based on reencrypting is discarded. In this case, it is possible to factor n in a chosen-ciphertext attack. We describe the attack for EPOC-1 with parameters ℓ_M ℓ_R, such that $\ell_R + \ell_M = k - 1$ but the attack also applies in different contexts. One simply submits a ciphertext C computed as $f(x, H(x))$, where x is randomly chosen and $\simeq 2^k$. Since $x - p$ is of the appropriate size, it is parsed as $M||R$ and the corresponding M is returned. From there, we obtain an approximation of p, consisting roughly of the leading ℓ_M bits of p. Since $n = p^2 q$, an approximation of q follows. Now, methods of [9], allow to factor n if these approximations are good enough.

Our final comment relates to a rather unusual feature of EPOC-3: the decryption algorithm may accept ciphertexts that the encryption algorithm does not produce. To see this, observe that the first component C_1 of a ciphertext can be replaced by $\tilde{C_1} = C_1 y^n \bmod n$. If the fourth component c_4 is modified accordingly, decryption is similar. Now, $\tilde{C_1}$ does not necessarily belong to the subgroup generated by g and h, a situation that the encryption algorithm cannot produce. It does not seem that this unexpected feature has any consequence, in terms of security.

3 Security level of the cryptographic primitive

In this section, we investigate the security of the underlying cryptographic primitive, both in terms of complexity-theoretic reductions and with respect to the recommended parameters.

3.1 Complexity-theoretic arguments

Documents [11, 12] relate the security of the scheme to various computational problems related to factoring.

3.1.1 The p-subgroup assumption

The p-subgroup assumption appears in [28] and is an extension of quadratic residuosity and of higher residuosity. Our treatment differs from [28] in that we define the assumption as the statement that it is hard to distinguish the distributions $\mathbf{R_n}$ and $\mathbf{D_n}$, where n is an integer of the form p^2q, $\mathbf{R_n}$ consists of the uniform distribution on $\mathbb{Z}_n^\star$ and $\mathbf{D_n}$ of the uniform distribution on elements of $\mathbb{Z}_n^\star$ which are p-th powers. A quantitative version measures the maximum advantage $\mathsf{AdvpSG}(t)$ of a statistical test T that runs in time t. This means the maximum of the difference of the respective probabilities that T outputs 1, when probabilities are taken over $\mathbf{R_n}$ or $\mathbf{D_n}$.

The above definition appears a bit unusual, since it is not apparent how to generate $\mathbf{D_n}$: the obvious construction, based on the image of the uniform distribution by the function $x \longrightarrow x^p \bmod n$ requires knowledge of p. However, one can notice that raising to the power pq yields a permutation of the set of pth powers. This shows that $\mathbf{D_n}$ can also be generated as the image of the uniform distribution by the function $x \longrightarrow x^n \bmod n$.

The above shows that there is a standard self-reducibility argument: by randomization, it is possible to transform an arbitrary element x into an equivalent one xy^n i.e. the output is in $\mathbf{D_n}$, if and only if the input is. Thus, if $\mathsf{AdvpSG}(t)$ is significant, one can use a distinguisher to decide, with probability close to one, whether an integer is in $\mathbf{D_n}$. This involves performing repeated tests with the distinguisher and deciding whether the number of one outputs has a bias towards $\mathbf{D_n}$ or $\mathbf{R_n}$. Based on the law of large numbers, a decision with small constant error probability requires running $O(\mathsf{AdvpSG}^{-2})$ tests. One can decrease the error probability drastically by repeating the above computations an odd number of times and deciding based on the median of the averages. Thus, the loss in the reduction is huge. Thus, despite its elegance, the self-reducibility argument is a bit misleading in terms of exact security.

The variant of the p-subgroup assumption that appears in [28] is closely related to the semantic security of the cryptosystem in a chosen plaintext scenario. It simply asserts that, for the cryptosystem described in this paper, it is hard to distinguish

encryptions of zero from encryptions of one. Call this assumption the OU assumption, from the initials of the authors. We show that the OU assumption and the p-subgroup assumption are equivalent.

Theorem 2 *Let $\mathcal{A}$ be an algorithm that breaks the semantic security of the Okamoto-Uchiyama scheme for encryptions of 0 and 1, with advantage ε, within time bound t. There exists a machine $\mathcal{D}$ that is able to distinguish $\mathbf{D_n}$ from $\mathbf{R_n}$ with advantage ε', within time bound t', where*

$$\begin{aligned} \varepsilon' &\geq \varepsilon/2 - \frac{1}{p} - \frac{2^{2k+1}}{2^\ell} \\ t' &\leq t + \tau \end{aligned}$$

where ℓ is the bit-length of the random string r used to encrypt, and where the overhead τ accounts for performing computations with integers of size $|n|$ and is bounded by $\mathcal{O}(k^3)$.

Proof: We turn an algorithm $\mathcal{A}$ that breaks the semantic security for encryptions of 0 and 1, into a distinguisher $\mathcal{D}$ between $\mathbf{D_n}$ or $\mathbf{R_n}$, as follows:

1. Run the part of the key generation algorithm that produces p, q, n
2. Take the input to $\mathcal{D}$ as the element g of the public key (recall that h is $g^n \bmod n$)
3. Toss a coin b and generate a random r as prescribed by the cryptosystem; if $b = 0$, submit $y = h^r \bmod n$ as the challenge ciphertext to $\mathcal{A}$; otherwise submit $y = gh^r \bmod n$
4. Let b' be the answer of $\mathcal{A}$; return bit $b = b'$

Notice that, if g is chosen at random in $\mathbb{Z}_n^\star$, then, since $\mathbb{Z}_{p^2}^\star$ is cyclic of order $p(p-1)$, g^{p-1} is of order p with overwhelming probability $1 - \frac{1}{p}$. Based on this observation, we see that, when the input to $\mathcal{D}$ is from $\mathbf{R_n}$, the probability that $\mathcal{D}$ returns $b' = b$ as 1 is very close to $\frac{1}{2}(\theta_1 + (1 - \theta_0))$, where θ_0 be the probability that algorithm $\mathcal{A}$ outputs 1 on inputs which are ciphertexts of 0, and θ_1 the probability that it outputs 1 when they are ciphertexts of 1. By very close, we mean that the difference is bounded by $\frac{1}{p}$. On the other hand, we claim that, when the input to $\mathcal{D}$ is from $\mathbf{D_n}$, the probability that $\mathcal{D}$ returns $b' = b$ as 1 is $\simeq 1/2$. To see this, we have to study more carefully the distributions on $\mathbb{Z}_n^\star$ generated by step 2 above. We use chinese remaindering and let γ_0 and γ_2 be generators of $\mathbb{Z}_{p^2}^\star$ and $\mathbb{Z}_q^\star$ respectively. Let $\gamma_1 = \gamma_0^p \bmod p^2$. A randomly chosen element from $\mathbf{D_n}$ corresponds to a pair $(\gamma_1^{u_1}, \gamma_2^{u_2})$ and, taking logarithms, we have to study the following two distributions

$$(rnu_1 \bmod (p-1), rnu_2 \bmod (q-1))$$

and

$$((rn+1)u_1 \bmod (p-1), (rn+1)u_2 \bmod (q-1))$$

Note that n is prime to $(p-1)(q-1)$, so that one can consider the inverse α of n modulo $(p-1)(q-1)$. The second distribution becomes

$$((r+\alpha)nu_1 \bmod (p-1), (r+\alpha)nu_2 \bmod (q-1))$$

and differs from the first one on a set of at most 2^{2k+1} elements. If r ranges over a set of size 2^ℓ, with ℓ much larger than 2^{2k+1}, we conclude that the distributions on $\mathbb{Z}_n^\star$ generated by step 2 are extremely close and that b' is therefore independent of b, except on a set of probability $\leq \frac{2^{2k+1}}{2^\ell}$.

We now turn to the converse of the theorem.

Theorem 3 *Let $\mathcal{D}$ be a machine that is able to distinguish $\mathbf{D_n}$ from $\mathbf{R_n}$ with advantage ε, within time bound t. There is an attacker $\mathcal{A}$ breaking the semantic security of the Okamoto-Uchiyama scheme form [28] for encryptions of 0 and 1, with advantage ε', within time bound t', with*

$$\begin{aligned} \varepsilon' &\geq \varepsilon - \frac{1}{p} \\ t' &\leq t + \tau \end{aligned}$$

where ℓ is the bit-length of the random string r used to encrypt and where the overhead τ accounts for performing computations with integers of size $|n|$ and is bounded by $\mathcal{O}(k^3)$.

The proof is straightforward, using the random self-reducibility: one simply handles $cy^n \bmod n$, where c is the challenge ciphertext and y is a random element of $\mathbb{Z}_n^\star$. It is immediate to see that ciphertexts of 0 generate $\mathbf{D_n}$ by randomization and ciphertexts of 1 generate the subset of $\mathbf{R_n}$ consisting of elements which are not p-th powers. Since this differs from $\mathbf{R_n}$ by a set of probability $\leq \frac{1}{p}$, the result follows.

Thus, we have shown that our version of the p-subgroup assumption, of a more structural character, is essentially equivalent to the version in [28]. However, the same is not true of the version of the p-subgroup assumption that appears in the submission. Again, the basic statement on which security is based is that, for the cryptosystem described in [11], it is hard to distinguish encryptions of zero from encryptions of one. This assumption cannot hold. To see why, notice that, since $n = p^2q$, the Jacobi symbol indicates whether an element is a square modulo q or not. Now, h is chosen randomly and thus, is a square modulo q with probability 1/2. Similarly, g is a non residue modulo q with probability 1/2. If both events happen, then ciphertexts of 0 have Jacobi symbol 1, while ciphertexts of 1 have Jacobi symbol -1. At this point, one may wonder whether the p-subgroup assumption has been the subject of enough research to build a cryptosystem.

3.1.2 The computational p-subgroup assumption

Related to the above is another assumption which we call the computational p-subgroup assumption. It states that, given y and g, it is hard to compute the unique element x, $0 \leq x < p$, such that $g^{-x}y$ is a p-th power. Uniqueness follows from the trapdoor computation of x as $\frac{L(y^{p-1} \bmod n)}{L(g_p)}$. It is obvious that the p-subgroup assumption is a stronger assumption than its computational version. Furthermore, it is easily seen that the latter is equivalent to factoring: in order to factor n using an algorithm that solves the computational p-subgroup problem, one simply submits $g^x y^n$, where x is slightly larger than p. The algorithm returns $x \bmod p$, from which we obtain a small multiple of p. Using arguments similar to those in theorem 2, one can see that the computational p-subgroup assumption is essentially equivalent to inverting the Okamoto-Uchiyama cryptosystem from [28]. We simply show how to factor, given an inversion algorithm.

Theorem 4 *If a machine $\mathcal{A}$ is able to compute x from $f(x,r)$, $x \in \{0,1\}^{k-1}$, with probability ε, within time bound t, there exists a machine $\mathcal{B}$ that is able to factor n with probability $\varepsilon' \geq \varepsilon/2(1-\frac{1}{2^{k-2}})(1-\frac{2^{2k+1}}{2^\ell})$, within time bound $t' \leq t + \tau$, where the overhead τ accounts for performing computations with integers of size $|n|$ and is bounded by $\mathcal{O}(k^3)$.*

Proof: Let us consider an adversary $\mathcal{A}$ able to compute x from $y = f(x,r)$ with probability ε, within time bound t:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[x \stackrel{R}{\leftarrow} \{0,1\}^{k-1}, y \leftarrow f(x,r), z \leftarrow \mathcal{A}(y) : z = x] > \varepsilon$$

where probabilities include r as well as the internal random tape of the probabilistic machine $\mathcal{A}$.

We denote by F the following function: for any $x \in \mathbb{Z}_n$, $F(x) \leftarrow g^x \bmod n$. we claim that F essentially generates the same distribution as f. To see this, write the order of g as pd, where d divides $(p-1)(q-1)$ and therefore is prime to p. Elements of the subgroup generated by g can be represented by their discrete logarithm in base g and, using Chinese remaindering, this means a pair (u,v), with $u \in \mathbb{Z}_p$ and $v \in \mathbb{Z}_d$. When x ranges over $\mathbb{Z}_n$, then, discarding elements $> p(p-1)(q-1)$ yields a uniform distribution of (u,v). This leaves aside a proportion $\leq \frac{1}{p} + \frac{1}{q} \leq \frac{1}{2^{k-2}}$. Now, $f(x,r)$ yields pairs $(x, x+nr \bmod d)$ and, since n is prime to d, the second coordinate differs from the uniform distribution by a multiplicative factor bounded from below by $1 - \frac{2d}{2^\ell}$, where ℓ is the bit-length of r. This is $\geq 1 - \frac{2^{2k+1}}{2^\ell}$. To conclude, note that a randomly chosen element (u,v) according to the first distribution appears in the second if $u < 2^{k-1}$ i.e. with probability at least $1/2$. If this happens, $\mathcal{A}$ returns $x \bmod p$, hence a multiple μ of p , which is $< pq$. Computing the gcd of μ and n allows to factor.

Observe that it is not straightforward to adapt the above proof to the cryptosystem described in the submission.

We conclude the section by mentioning that no other way of solving either the p-subgroup problem or the computational p-subgroup problem is known besides factoring n.

3.2 Size of the parameters

As was just observed the security of the basic Okamoto-Uchiyama scheme on which EPOC relies is related to the hardness of the p-subgroup problem and factoring is the only known way of solving this problem. Thus, we have to study the hardness of factoring integers n of the form $n = p^2q$. It is unclear whether or not factoring is easier for such numbers than it is in case of integers with two prime factors. In order to state an opinion, we briefly review the performances of known factoring algorithms. Such algorithms fall into three families, according to their sensitivity to the size of the factors and to the existence of repeated factor.

3.2.1 Factoring techniques sensitive to the size of the smaller factor

Pollard's ρ-method. The idea behind the method is to iterate a polynomial P with integer coefficients, that is to say computing $x_1 = P(x_0)$, $x_2 = P(P(x_0))$, etc. In time complexity $O(\sqrt{p})$, where p is the smallest prime factor of n, one finds a collision modulo p, i.e. two values x_i and x_j, $i \neq j$, such that $x_i = x_j \bmod p$. Computing $\gcd(x_i - x_j, n)$ factors n.

Although there are several optimizations, the ρ-method can only be used to cast out "small" factors of an integer (say 30-digit factors). As far as we know, it has not been used to find significantly larger factors.

The $p-1$ method. Let B be a positive integer. A number is B-smooth if it is a product of prime numbers all $< B$. B-smooth numbers are usually used through a table of primes $< B$. The $p-1$ method relies on the use of Fermat's little theorem: if $p-1$ is B-smooth, then the computing $\gcd(n, a^{\ell(B)} - 1)$ factors n, where $\ell(B)$ is the product of all prime factors $< B$.

The security against this factoring method is usually addressed by the requirement that each of $p-1$, $q-1$ has a large prime factor. However, such requirement does not appear in document [11].

The elliptic curve method. The ECM is a generalization of the $p-1$ method, for which the above simple countermeasure is not sufficient. Consider an elliptic curve mod$n$ with equation

$$y^2 = x^3 + ax + 1$$

If the number of points of this curve modulo p is B-smooth, then a factor of n can be discovered along the computation of the scalar multiplication of $M_0 = (0, 1)$ by $\ell(B)$, according to the group law of the elliptic curve.

The success probability of the algorithm is as follows: Let

$$L(x) = \exp(\sqrt{\ln x \ln \ln(x)})$$

Then, the curve is $L(p)^\alpha$-smooth with probability $L(p)^{-1/(2\alpha)+o(1)}$. This is minimal for $\alpha = 1/\sqrt{2}$ and gives an expected running time of $L(p)^{\sqrt{2}+o(1)}$ group operations on the curve.

There have been several improvements of ECM factoring, notably the FFT extension of P. Montgomery. Furthermore, several implementations of ECM are available. The current ECM factoring record was established in December 1999 when a prime factor of 54 digits of a 127-digit composite number n was found with GMP-ECM, a free implementation of the Elliptic Curve Method (see [23]). The limit used was $B = 15,000,000$.

In a recent paper [7], Richard Brent extrapolates the ECM record to be of D digits at year about

$$Y = 9.3 * \sqrt{D} + 1932.3$$

this would give records of $D = 60$ digits at year $Y = 2004$ and $D = 70$ at year 2010. Such record would need $B \simeq 2,900,000,000$ and require testing something like 340,000 curves. The behavior of ECM on integers of the form p^2q has not been documented through experiments. There are indications that it is possible to slightly speed up ECM for numbers of the form p^2q (see [29]). Still, given Brent's prediction, it is highly unlikely that the ECM method can endanger the scheme in a foreseeable future.

3.2.2 Factoring techniques which are not sensitive to the size of the smaller factor

Quadratic sieve. The quadratic sieve method (QS) factors n by gathering many congruences of the form

$$x^2 = (-1)^{e_0} p_1^{e_1} \cdots p_m^{e_m}$$

where $p_1, \cdots, p_m$ is a list of prime numbers $< B$, called the factor base. This is done by finding B-smooth numbers of the form $Q(a) = (\sqrt{n}+a)^2 - n$. It turns out that there is a very efficient sieving process that performs the job without division, hence the name QS. Once enough congruences have been gathered, one obtains another congruence of the same type with all exponents e_i even: this is done by Gaussian elimination mod2. Thus we get some relation $x^2 = y^2 \bmod n$ and, with significant probability, computing $\gcd(x-y, n)$ factors n. The time complexity of QS is $O(L(n)^{1+o(1)})$ but, as it uses very simple operations, it is usually more efficient than ECM for numbers whose smallest prime factor exceeds $n^{1/3}$.

Many improvements of the basic method have been found, notably the multiple polynomial variation (MPQS) and the large prime variation. This has led to very efficient implementation and, until the mid-nineties, was used to set up factoring records. The largest number factored by MPQS is the 129-digit number from the "RSA Challenge" (see [31]). It was factored in April 1994 and took approximately 5000 mips-years (see [2]).

Number field sieve. The number field sieve (NFS) is somehow similar to the QS but it searches for congruences in some number field (algebraic extension of the rational numbers). The method uses two polynomials with a common root m modulo n. These polynomials should have as many smooth values as possible. The time complexity of NFS is

$$O(e^{(\ln n)^{1/3}(\ln \ln n)^{2/3}(C+o(1))})$$

for a small constant C (about $(64/9)^{1/3} \simeq 1.923$). This is asymptotically considerably better than QS. In practical terms, NFS beats QS for numbers of more than about 110 digits (see [10]). The number field sieve was used to factor the 130-digit RSA challenge number in April 1996, with an amount of computer time which was only a fraction of what was spent on the old 129-digit QS-record. It was later used to factor RSA-140 in February 1999 with an amount of computer time about 2000 Mips-years. In August 1999, the factorization of RSA-155 from the RSA list was obtained ([8]). The amount of computer time spent on this new factoring world record is equivalent to 8000 mips-years, whereas extrapolation based on RSA-140 and the asymptotic complexity formula for NFS predicted approximately 14000 mips-years. The gain was caused by an improved polynomial search method. The final linear algebra part took 224 CPU hours and 2 Gbytes of central memory on a Cray C916.

The main obstacle to a fully scalable implementation of NFS is actually the linear algebra, although progress has been made (see [24]). In [8], the authors derive the following formula

$$Y = 13.24D^{1/3} + 1928.6$$

for predicting the calendar year for factoring D-digit number by NFS. The same formula appears in [7] and produces $Y = 2016$ for $D = 289$, i.e. for a 960 bit modulus, proposed as a minimum choice in EPOC.

3.2.3 Specific factoring techniques for numbers of the form $p^d q$

In recent work (see [6]), a new factoring method that applies to integers of the form $p^d q$ has been found. The method is based on an earlier result of Coppersmith (see[9]), showing that an RSA modulus $n = pq$, with p, q of the same size, can be factored given half the most significant bits of p. It turns out that, for numbers of the form $n = p^d q$, with p, q of the same size, fewer bits are needed.

Note that disclosing the leading bits of p provides a rough approximation P of p. What remains to be found is the difference $x = p - P$. The new method is based on finding polynomials with short enough integer coefficients, which vanish at x modulo some power p^{dm} of p. Such polynomials are actually zero at x. Thus, factoring is achieved by finding the appropriate root. The polynomial itself is computed by the LLL lattice reduction algorithm from [21]. LLL is run on lattices of dimension d^2 with basis vectors of size $O(d \log n)$. Let γ be the corresponding computing time. Taking into account the workfactor tied with guessing the approximation of p, the total running time is

$$2^{\frac{c+1}{d+c} \cdot \log p} . \gamma$$

where c is such that $q \simeq p^c$.

Comparing the above estimate with the running time for ECM, one can see that the new method beats ECM for d larger than, approximately $\sqrt{\log p}$. In the case of EPOC, where $d = 2$, the algorithm is certainly impractical.

3.3 Conclusion

Document [11] does not provide a range of parameters for the cryptosystem. It simply mentions that k should be at least 320. This yields a bit-size 960 for n, which is 289 digits. Based on current estimates, such minimal parameters appear secure for only fifteen years or so. Although predictions should be taken with great care, this means a rather short "lifetime". Of course, one can increase the size of the parameters. However, parts of the basic security arguments rely on having the size of r larger than $2k$. Since the suggestion for this bit-size is 832, larger values of k make some of the security claims invalid. The main concern with the cryptosystem is whether the p-subgroup assumption has been the subject of enough research to build a cryptosystem. The incorrect version of the assumption included in the submission tends to indicate that the answer is no.

4 Security Analysis of EPOC-1

Document [11] includes several security statements and points to the literature, notably [16] for details.

4.1 Formal framework

An asymmetric encryption scheme is *semantically secure* if no polynomial-time attacker can learn any bit of information about the plaintext from the ciphertext, except its length. More formally, an asymmetric encryption scheme is (t, ε)-IND where IND stand

for *indistinguishable*, if for any adversary $\mathcal{A} = (A_1, A_2)$ with running time bounded by t, the advantage

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \Pr_{\substack{b \stackrel{R}{\leftarrow} \{0,1\} \\ r \stackrel{R}{\leftarrow} \Omega}} \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{K}(1^m), (M_0, M_1, st) \leftarrow A_1(\mathsf{pk}) \\ C \leftarrow \mathcal{E}_{\mathsf{pk}}(M_b; r) : A_2(C, st) \stackrel{?}{=} b \end{array} \right] - 1/2$$

is $< \varepsilon$, where the probability space includes the internal random coins of the adversary, and M_0, M_1 are two equal length plaintexts chosen by the adversary in the message-space $\mathcal{M}$.

Another security notion has been defined in the literature, the so-called *non-malleability* [14]. Informally is states that it is impossible to derive, from a given ciphertext, a new ciphertext such that the plaintexts are meaningfully related. We won't discuss this notion any further since it has been proven equivalent to semantic security in an extended attack model.

The above definition of semantic security covers passive adversaries. It is a *chosen–plaintext* or CPA attack since the attacker can only encrypt plaintext. In the extended model, the *adaptive chosen–ciphertext* or CCA attack, the adversary is given access to a decryption oracle and can ask the oracle to decrypt any ciphertext, with the only restriction that it should be different from the challenge ciphertext. It has been proven in [3] that, under CCA, semantic security and non-malleability are equivalent. This is the strongest security notion currently considered.

4.2 Chosen ciphertext security of EPOC-1

We turn to the security analysis. A first observation is that the p-subgroup assumption implies the semantic security of the basic Okamoto-Uchiyama scheme from [28], in the CPA scenario. Indeed, a CPA attacker $\mathcal{A} = (A_1, A_2)$ can be converted into a distinguisher for encryptions of zero and one: one simply handles A_2 the modified challenge $g^{m_0} c^{m_1 - m_0} g^{rn}$, where m_0 and m_1 are the output test messages of A_1 and c is the queried $\{0,1\}$-encryption. The additional term g^{rn} is for randomization and ensures, provided $\frac{2^{2k}}{2^\ell}$ is small, that one gets a uniformly distributed ciphertext of $m_0 \cdot (1-b) + b \cdot m_1$. The arguments are similar to those of theorems 2 and 4. The advantage decreases by a minute term $\simeq \frac{2^{2k}}{2^\ell}$.

We now want to prove that EPOC-1 is IND-CCA in the random oracle model, based on the assumption that the p-subgroup assumption holds. More precisely, we wish to establish the following exact security result.

Theorem 5 *Let $\mathcal{A}$ be a CCA–adversary attacking $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, within time bound t, with advantage ε, making q_D and q_H queries to the decryption oracle and the hash function, respectively. Then there exists an adversary $\mathcal{B}$ attacking the semantic security of the*

basic Okamoto-Uchiyama scheme with advantage ε' and within time bound t', where

$$\begin{aligned} \varepsilon' &\geq \varepsilon - \frac{2q_H}{2^{\ell_R}} - \frac{q_D}{d} \\ t' &\leq t + q_H \cdot \tau \end{aligned}$$

where ℓ_R is the bisize of R, d the order of h and τ is the time needed to execute the encryption algorithm and is bounded by $\mathcal{O}(k^3)$

The scheme can actually be proven "plaintext–aware" [5, 3], which is claimed to imply chosen-ciphertext security. To prove the above, we turn $\mathcal{A}$ into an attacker for the OU scheme. This requires simulating the oracle and describing a plaintext-extractor.

4.2.1 Description of the reduction

Let $\mathcal{A} = (A_1, A_2)$ be an adversary against the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The description of $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ first runs $\mathcal{K}(1^k)$ to obtain the public key
2. next, $\mathcal{B}$ runs A_1 on the public data, and gets a pair of messages $\{M_0, M_1\}$ as well as a state information s. It chooses two random strings R_0, R_1 of the appropriate length, then flips a random bit b and then defines $C \leftarrow y$, to be a ciphertext of $(M_b||R_b)$
3. $\mathcal{B}$ runs $A_2(C, s)$ and returns the answer b' of $A_2(C, s)$.

Of course, during the entire simulation, $\mathcal{B}$ also has to simulate answers from the random oracle. This is done using the **H-list**, a dynamic data structure consisting of all queries to the random oracle H together with the respective answers. For a fresh query h to H, i.e. a query not on the **H-list**, the oracle picks a random answer. Finally, we define the plaintext-extractor as follows:

1. on a query C' to the decryption oracle, $\mathcal{B}$ looks at the **H-list**. If there is an element on this list which is parsed as $(M'||R', H(M'||R'))$, if it is of the appropriate bit-length and if $f(M'||R', H(M'||R')) = C'$, then $\mathcal{B}$ outputs M' as the requested plaintext
2. Otherwise, $\mathcal{B}$ rejects the ciphertext

4.2.2 Probabilistic analysis

We wish to compare the behavior of several games

1. game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle, and the random oracle is perfect

2. game $\mathcal{G}_2$, which is as $\mathcal{G}_1$ but aborts without calling the random oracle, whenever some $(M_i||R_i)$ appears as a fresh query. In this case the output b' is set at $b' = i$. The decryption queries are still answered by the actual decryption oracle.

3. game $\mathcal{G}_3$, where the random oracle is simulated

4. game $\mathcal{G}_4$, which is as $\mathcal{G}_3$ but where the decryption queries are answered by the plaintext-extractor

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is exactly the advantage ε of $\mathcal{A}$. We relate the probabilities of the same event in all games repeatedly using the simple yet useful lemma from [32]

Lemma 1 *Let E, F, and E', F' be events of two probability spaces such that both*

$$\Pr[E|\neg F] = \Pr[E'|\neg F'] \text{ and } \Pr[F] = \Pr[F'] \leq \varepsilon.$$

Then,

$$|\Pr[E] - \Pr[E']| \leq \varepsilon$$

Proof: We write

$$\Pr[E] = \Pr[E|\neg F]\Pr[\neg F] + \Pr[E|F]\Pr[F]$$
$$\Pr[E'] = \Pr[E'|\neg F']\Pr[\neg F'] + \Pr[E|F']\Pr[F']$$

Hence

$$\Pr[E] - \Pr[E'] = \Pr[E|\neg F](\Pr[\neg F] - \Pr[\neg F']) + (\Pr[E|F]\Pr[F] - \Pr[E|F']\Pr[F'])$$

The right hand side becomes $\Pr[E|F]\Pr[F] - \Pr[E|F']\Pr[F']$, which is bounded by ε.

Game $\mathcal{G}_2$ differs from $\mathcal{G}_1$ if it aborts. Observe that the input C is a ciphertext of $M_b||R_b$, but this leaves the choice of R_{1-b} random. Thus, R_{1-b} can appear in a query to H only with probability $\frac{q_H}{2^{\ell_R}}$. If this does not happen, then a query of the form $M_i||R_i$ is such that $i = b$. Thus the advantage of $\mathcal{G}_2$ exceeds $\varepsilon - \frac{q_H}{2^{\ell_R}}$.

The simulation in game $\mathcal{G}_3$ is perfect unless it conflicts with the implicit constraint coming from the assumption that C is a ciphertext of $(M_b||R_b)$. In the find stage, R_b has not been chosen and this happens, for each oracle query, with probability $\leq \frac{1}{2^{\ell_R}}$. In the guess stage, this cannot happen since $\mathcal{G}_2$ has earlier aborted. Altogether, we discard a set of probability $\leq \frac{q_H}{2^{\ell_R}}$. Applying the lemma, we see that the probabilities in the resulting games differ by at most this value.

To go from $\mathcal{G}_3$ to $\mathcal{G}_4$, we have to bound the probability that the plaintext-extractor rejects a correct ciphertext C'. Let $M'||R'$ be the decryption of C' under the OU scheme. The situation that we wish to avoid happens if the value of H at $(M'||R')$ is later set at a value r' such that $C' = f(M'||R', r')$. Let d be the order of h in $\mathbb{Z}_n^\star$. We get

$$h^{r'} = C' \cdot g^{-(M'||R')} \bmod n$$

which defines a subset of probability $\frac{1}{d}$. Altogether, we get the bound $\frac{q_D}{d}$.

Finally, in game $\mathcal{G}_4$, the value of $H(M_b||R_b)$ is left random. Thus $\mathcal{G}_4$ can be viewed as a distinguisher for the OU scheme. Its advantage is at least

$$\varepsilon - \frac{q_H}{2^{\ell_R}} - \frac{q_H}{2^{\ell_R}} - \frac{q_D}{d}$$

Now, the heavy load in the simulation is reencryption of all candidates from the **H-list**. This means q_H such computations. This completes the proof.

4.3 Summary of the security analysis

Piecing together the results of theorems 5 and 3, we see that EPOC-1 is semantically secure against CCA adversaries, based on the p-subgroup assumption, provided that it is derived from the version of the Okamoto-Uchiyama cryptosystem described in [28]. We also note that the sequence of reductions under this result can be made tight by an appropriate choice of the parameters.

Since the version of the p-subgroup assumption given in the submission does not appear correct, we do not see how to extend the security proof in this case. Also, the submission does not make completely clear the role of the various parameters: from the analysis in section 5, we see that

1. the bit-length ℓ of the hash function should be much larger than $2k$.
2. the order of h should be large

No part of the submission specifically points out these requirements.

5 Security Analysis of EPOC-2

Document [11] includes several security statements and points to the literature, notably [17] for details.

5.1 Semantic security of key encapsulation

Before we turn to the actual security analysis, we would like to focus on what can be called *key encapsulation*, a term introduced in [32], but which we use here with a slightly improper meaning, due to the fact that, in the present context, the session key is message dependent. This analyzes the semantic security of the symmetric session key itself in the context of CCA-adversaries: the attacker is allowed to query a decryption oracle by submitting an element C_1 of $\mathbb{Z}_n^\star$ and receiving the key session key $\mathsf{k} = G(R)$ as the answer, where C_1 is $f(R, H(M||R))$ as prescribed by the cryptosystem. His aim is to distinguish the distribution consisting of a ciphertext C_1 and the corresponding key material k from the analogous distribution where the key material is replaced by a random string. We denote by $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t, q_D)$ the maximal advantage for any adversary in distinguishing both distributions within time bound t, after q_D queries to the decryption oracle.

We prove the following:

Theorem 6 *Let $\mathcal{A}$ be a CCA–adversary attacking the key encapsulation scheme of EPOC-2 within time bound t, with advantage ε, making q_D, q_G and q_H queries to the decryption oracle and the hash functions G and H, respectively. There exists an adversary $\mathcal{B}$ inverting the Okamoto-Uchiyama scheme with advantage $\frac{\varepsilon'}{q_H+q_G}$ and within time bound t', where*

$$\begin{aligned} \varepsilon' &\geq \varepsilon/2 - \frac{q_G + q_H}{2^{\ell_R}} - \frac{q_D}{d} \\ t' &\leq t + q_H \cdot \tau \end{aligned}$$

ℓ_R is the bit-size of R, d the order of h and τ is the time needed to execute the encryption algorithm and is bounded by $\mathcal{O}(k^3)$

Proof: From $\mathcal{A}$, we build a machine $\mathcal{B}$ which attempts to invert the OU scheme.

1. $\mathcal{B}$ receives its input, an OU ciphertext C
2. $\mathcal{B}$ tosses a random coin b. If $b = 0$ it generates a random session key k and submits the pair (C, k) to the distinguisher $\mathcal{A}$; if $b = 1$, it manufatures a correct pair (C', k') as prescribed by the scheme. In both cases, it handles the pair to $\mathcal{A}$
3. when the execution of the distinguisher has ended, $\mathcal{B}$ outputs a tentative the decryption of C, from the queries asked to G and H (see below)

Of course, during the entire simulation, $\mathcal{B}$ also has to simulate answers from the random oracle. This is done using a **H-list** and a **G-list**. For a fresh query h to H, i.e. a query not on the **H-list**, the oracle picks a random answer and similarly for G. We also define a plaintext-extractor as follows:

1. on a query C' to the decryption oracle, $\mathcal{B}$ looks at the **H-list**. If there is an element on this list which is parsed as $(M'||R', H(M'||R'))$, with R' of the appropriate length, and if $f(R', H(M'||R')) = C'$, then $\mathcal{B}$ outputs $G(R')$ as the requested session key, including it on the **G-list**, if needed.

2. Otherwise, $\mathcal{B}$ rejects the ciphertext

Finally, we explain what is the final answer of $\mathcal{B}$ is, besides the b' bit computed by $\mathcal{A}$. When execution has ended, $\mathcal{B}$ looks at the **G-list** and the **H-list** and outputs a randomly chosen element, which appears as a question R in the **G-list** or such that $(M||R)$ appears as a question in the **H-list**.

We wish to compare the behavior of several games

1. game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle, and the random oracle is perfect

2. game $\mathcal{G}_2$, where the decryption queries are answered by the plaintext-extractor

3. game $\mathcal{G}_3$, where the oracle is simulated

We let R be the decryption of C under the OU scheme and we denote by ε' the probability that R appears in the **G-list** or the **H-list** upon completion.

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is $\frac{1}{2}(\theta_1 + (1 - \theta_0))$, where θ_0 be the probability that algorithm $\mathcal{A}$ outputs 1 on inputs taken from pairs (C, k), with random session key k and θ_1 the probability that it outputs 1 when they are taken from pairs constructed according to the rules of key-encapsulation. This is $1/2 + \frac{\varepsilon}{2}$.

To go from $\mathcal{G}_1$ to $\mathcal{G}_2$we have to bound the probability that the plaintext-extractor rejects a correct ciphertext C'. Let R' be the decryption of C' under the OU scheme. The situation that we wish to avoid happens if the value of H at $(M'||R')$ is later set at a value r' such that $C' = f(R', r')$. Let d be the order of h in $\mathbb{Z}_n^\star$. We get

$$h^{r'} = C' \cdot g^{-(M'||R')} \bmod n$$

which defines a subset of probability $\frac{1}{d}$. Altogether, we get the bound $\frac{q_D}{d}$.

The simulation in game $\mathcal{G}_3$ is perfect unless it conflicts with the implicit constraint on H coming from the assumption that C is a ciphertext corresponding to an unknown (but fixed) R. In the find stage, R has not been chosen and is involved in a G-query or a H-query with probability $\leq \frac{1}{2^{\ell_R}}$. In the guess stage, this can only happen when R does appear as a possible choice for the final output of $\mathcal{B}$. Altogether, we discard a set of probability $\leq \frac{q_G + q_H}{2^{\ell_R}} + \varepsilon'$. Applying the lemma, we see that the probabilities in the resulting games differ by at most this value. Note that, once the exceptional runs have been discarded, we are left with executions where R is not asked to G. We can

freely interpret $G(R) = \mathsf{k}$ and are left with a distribution of inputs independent from b. The probability that $b' = b$ in this case is therefore $1/2$.

Finally, we have bounded the probability that $\mathcal{B}$ inverts the OU cryptosystem by $\frac{\varepsilon'}{q_H+q_G}$, where ε' is such that

$$\varepsilon/2 \leq \varepsilon' + \frac{q_D}{d} + \frac{q_G + q_H}{2^{\ell_R}}$$

Now, the heavy load in the simulation is the reencryption of all candidates from the **H-list**. This means q_H such computations.

5.2 From key encapsulation to chosen ciphertext security

We prove the following.

Theorem 7 *Let $\mathcal{A}$ be a CCA–adversary attacking the hybrid cryptosystem, within time bound t, with advantage ε, making q_D queries to the decryption oracle. Then*

$$\begin{aligned} \varepsilon &\leq \mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D) + \mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t') + \frac{q_D}{d} \\ \textit{where } t' &\leq t + \mathcal{O}(q_D) \end{aligned}$$

d the order of h, and $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$ denotes the security level of the symmetric encryption scheme, as defined below.

Before going further with the proof, let us define more formally the relevant security notion for the symmetric encryption.

5.2.1 Symmetric Encryption

A symmetric encryption scheme with key-length k, operating on messages of length ℓ, consists of two algorithms (E, D) which depends on a k-bit string k, called the secret key:

- the encryption algorithm $\mathsf{E}_{\mathsf{k}}(m)$ outputs a ciphertext c corresponding to the plaintext $m \in \{0,1\}^{\ell}$, in a deterministic way;
- the decryption algorithm $\mathsf{D}_{\mathsf{k}}(c)$ recovers the plaintext m associated to the ciphertext c.

A security notion similar to those used for asymmetric encryption is considered, known as *semantic security* [19]: a symmetric encryption scheme is *semantically secure* if no polynomial-time attacker can learn any bit of information about the plaintext from the ciphertext, besides its length. More formally, a symmetric encryption scheme

is (t, ε)-IND if, for any adversary $\mathcal{A} = (A_1, A_2)$ with running time bounded by t, $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) < \varepsilon$, where

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \Pr_{\substack{\mathsf{k} \xleftarrow{R} \{0,1\}^k \\ b \xleftarrow{R} \{0,1\}}} [(M_0, M_1, s) \leftarrow A_1(k), c \leftarrow \mathsf{E}_{\mathsf{k}}(M_b) : A_2(c, s) \stackrel{?}{=} b] - 1/2,$$

In the above, probabilities include the random coins of the adversary, and M_0, M_1 are two identical-length plaintexts in the message-space $\{0,1\}^\ell$.

We denote by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t)$ the maximal advantage of any adversary, against the semantic security of the scheme (E, D), within time bound t.

5.2.2 Security of EPOC-2

We define the attacker $\mathcal{A} = (A_1, A_2)$ and use this adversary as usual.

1. run the key generation algorithm for the encapsulation scheme
2. next run A_1 on the public data to get a pair of messages $\{M_0, M_1\}$ as well as a state information st. Choose a random bit b, run the key encapsulation scheme to get C_1 and the session key k, and encrypt M_b as C_2, using the session key
3. run $A_2((C_1, C_2), st)$ and get an answer b'. Finally, output bit $b = b'$.

As in the proof of theorem 6, we will envision several games:

- game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle
- game $\mathcal{G}_2$, where the session key to encrypt the test message M_b is replaced by a random string and where queries whose initial part matches with C_1 are decrypted using the same random string
- game $\mathcal{G}_3$, which is as $\mathcal{G}_2$ but where all queries whose initial part matches with C_1 are rejected

Note that the running time of all games is $t' = t + \mathcal{O}(q_D)$, where the second term accounts for the symmetric decryptions. Also, observe that the probability that $\mathcal{G}_1$ outputs 1 (which means $b' = b$) is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary operating within the real-life setting. As in the proof of theorem 6, we bound the difference of probabilities between $\mathcal{G}_1$ and $\mathcal{G}_3$.

In order to bound the difference between $\mathcal{G}_1$ and $\mathcal{G}_2$, observe that both can be played by calling the decryption oracle for key encapsulation rather than the actual decryption oracle. Game $\mathcal{G}_1$ needs as an additional input the session key corresponding to C_1 and,

similarly, game $\mathcal{G}_2$ will need the random key material. Thus, we have obtained a distinguisher between the distribution consisting of a ciphertext of M_b and the corresponding session key (in game $\mathcal{G}_1$) and the analogous distribution where the key is replaced by a random string (in game $\mathcal{G}_2$). This bounds the difference by $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$.

Going from game $\mathcal{G}_2$ to game $\mathcal{G}_3$, we note that a difference only occurs when one rejects a valid ciphertext (C_1, C_2'). Let M' be the corresponding plaintext. Since the queried ciphertext is different from the challenge ciphertext, M' is different from M. However, since the first component C_1 is similar, we have

$$f(R, H(M||R)) = f(R, H(M'||R))$$

where R denotes the random string used to generate C_1. The above yields the collision

$$h^{H(M||R)} = h^{H(M'||R)}$$

Since H is a random oracle, such collision appears with probability $\frac{1}{d}$, where d is the order of h.

To conclude, consider the probability of that $\mathcal{G}_3$ outputs one. In this game, a random string is drawn as a session key and used to encrypt a randomly chosen test message M_b under this key. The adversary outputs one if he has correctly guessed bit b. This is exactly the situation of a semantic distinguisher as defined in section 5.2.1. Therefore, the advantage of the adversary in this latter game $\mathcal{G}_3$ is bounded by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$.

Remark. There is a slight ambiguity under the notation $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$ in the above theorem: it actually refers to the advantage of a distinguisher for the key encapsulation scheme, which first runs A_1 and next uses as message space the two test messages output by A_1. It is easily seen that theorem 6 can be applied to such distinguishers. Thus, despite the ambiguity, we have chosen to split the security analysis, for the sake of clarity.

5.3 Summary of the security analysis

Piecing together the results of theorems 6, 7 and 4, we see that EPOC-2 is semantically secure against CCA adversaries, based on the hardness of factoring, if it is derived from the version of the Okamoto-Uchiyama cryptosystem described in [28]. The sequence of reductions under this result is not as good as in the case of EPOC-1, due to the multiplicative loss $\frac{1}{q_G+q_H}$ in theorem 6 and it is unclear whether it could lead to meaningful estimates.

There are technical difficulties to cover the general case, where h and g^p do not necessarily lie in the same subgroup. This comes from theorem 4, whose proof uses the fact that $g^x \bmod n$ essentially generates the same distribution as the cryptosytem. This is not true in general: for suitable choices of g and h, the quadratic residues modulo n appear with probability $1/2$ in the first case and $1/4$ in the second. One

could use $g^x h^r \bmod n$ in place of $g^x \bmod n$ but the analysis remains more subtle than expected: we see how to extend of theorem 4 when $\frac{p-1}{2}$, $\frac{q-1}{2}$, are mutually prime, with an extra multiplicative loss 1/2 in the probabilities, but have not attempted to cover the general case.

It can be noted that the hypotheses of the proof of theorem 4 requires that ℓ_R is $k-1$. A few bits can be discarded with a multiplicative loss 1/2 in the probabilities, for each bit. However, the submission only requires that $\ell_R \leq k-1$ and this leaves open the possibility that $\ell_R << k-1$, in which case the security proof collapses. The self-evaluation document [12], sets $\ell_R = k-1$.

Similarly, ℓ should be larger than $2k$. However, the submission does not include such requirement and this leaves open the possibility that $\ell << 2k$, in which case the security proof collapses. The self-evaluation document [12], sets $\ell = (2+c_0)k$.

Finally, the order d of h should be large. This is only implicit in the submission and the self-evaluation report.

Thus, as for EPOC-1, the submission does not make completely clear the role of the various parameters: again, the analysis shows that

1. the bit-length ℓ_R of the random string should be close enough to $k-1$ (the submission only requires $\ell_R \leq k-1$)

2. the bit-length ℓ of the hash function should be larger than $2k$

3. the order of h should be large

However, no part of the specification specifically points out these requirements, even if the results quoted in the self-evaluation report [12] duly consider cases where they hold.

6 Security Analysis of EPOC-3

Document [11] includes several security statements and an appendix, which is a preliminary version of [26, 27]. EPOC-3 is a hybrid cryptosystem which addresses the drawback that reencryption was needed at the decryption phase of EPOC-2 (as it was in EPOC-1). This is done by applying the conversion method from [26]. This conversion does not include the first hash function H at key encapsulation but uses it to as a tag, once the symmetric encryption has been performed. This is along the same lines as [1, 32]. However, the resulting scheme does not rely any more on factoring but on a version of the so-called gap-problems, which now discuss.

6.1 Gap higher residuosity

In the following, we review the version of the so-called gap problems (see [27]) that is needed in the current context. Besides the original definition, we consider a weaker assumption. We first define oracles, that an adversary can call. Notations are straightforward and come from 3.1.1.

- *a p-subgroup oracle*: on any input x from $\mathbb{Z}_n^\star$, it perfectly answers whether x comes from $\mathbf{D_n}$ or not.
- *a δ-p-subgroup oracle*: on any input x from $\mathbb{Z}_n^\star$, it answers whether x comes from $\mathbf{D_n}$ or not, with some error probability δ. More precisely, the advantage of this oracle is greater then $1-\delta$:

$$\left| \begin{array}{l} \Pr[oracle(x) = 1 \,|\, x \in \mathbf{D_n}] \\ -\Pr[oracle(x) = 0 \,|\, x \in \mathbf{R_n}] \end{array} \right| \geq 1 - \delta.$$

The reason why we introduce the second oracle is that, as already noted in section 3.1.1, if δ is significantly smaller than 1, one can use this δ-oracle ($\mathcal{O}((1-\delta)^{-2})$ times) to decide, with an error probability as small as wanted, whether an element is in $\mathbf{D_n}$ or not.

We now define the Gap higher residuosity problem, which consists in factoring a number of the form $n = p^2q$, having access to a p-subgroup oracle. A weaker assumption is the intractability of the δ-Gap higher residuosity problem, which consists in factoring a number of the form $n = p^2q$, having access to a δ-p-subgroup oracle.

The submission states that EPOC-3 is secure against CCA adversaries, provided the Gap higher residuosity problem is intractable and this can be extended to the formally weaker version of the hypothesis. It can be observed that both versions of the gap problem are polynomially equivalent (at least under a non-uniform reduction). Indeed, let $\mathcal{A}$ be an adversary that factors after q queries to a p-subgroup oracle, with probability ε (where q and $1/\varepsilon$ are polynomially bounded). Then, from any δ-p-subgroup oracle, one can build a δ'-p-subgroup oracle, achieving $\delta' < \varepsilon/2q$. Therefore, if one simulates the perfect oracle, called by $\mathcal{A}$, using this δ'-p- subgroup simulator, then $\mathcal{A}$ succeeds in solving the computational p-subgroup problem with probability $\varepsilon - q \times \delta' \geq \varepsilon/2$. Still, even if both problems are polynomially equivalent, the computational cost of the above reduction may be huge, depending on the original value of δ. In the following, we denote by $\mathsf{Succ}^{\mathsf{GHR}}(\delta, t, q)$ the maximal success probability of any adversary in factoring n within time t after less than q queries to a δ-p-subgroup oracle.

6.2 Semantic security of key encapsulation

As we did for EPOC-2, we would like to first focus on key encapsulation. However, this does not appear possible by lack of a correct simulator: when receiving a query C_1, the simulator should provide $G(R)$, where R is the decryption of C_1 under the Okamoto-Uchiyama scheme. A natural option is to search in the **G-list**, hoping to find the appropriate value of R, and to perform a check by calling the p-subgroup oracle. Although the approach is sound, the difficulty is to decide what to output when such R is not found: if the bit-length of R does not exceed ℓ_R, one could output a random value and reject otherwise. Unfortunately, the bit-length is unknown and there is no way to make a choice. Thus, we have to directly proceed to the full security proof.

6.3 Semantic security of EPOC-3

Theorem 8 *Let $\mathcal{A}$ be a CCA–adversary attacking the semantic security of EPOC-3 within time bound t, with advantage ε, making q_D, q_G and q_H queries to the decryption oracle and the hash functions G, H, respectively. There exists an adversary $\mathcal{B}$ inverting the Okamoto-Uchiyama scheme with the help of a δ-p-subgroup oracle, with advantage ε' and within time bound t', where*

$$\begin{aligned} \varepsilon &\leq \varepsilon' + \mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t') + \frac{q_D}{2^{\ell_H}} + (2q_H + q_G) \cdot \delta \\ \textit{where } t' &\leq t + \mathcal{O}(2q_H + q_G), \end{aligned}$$

and $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$ denotes the security level of the symmetric encryption scheme.

Proof: We start from an attacker $\mathcal{A} = (A_1, A_2)$ and use this adversary as always.

1. run the key generation algorithm for the encapsulation scheme
2. next run A_1 on the public data to get a pair of messages $\{M_0, M_1\}$ as well as a state information st. Choose a random bit b, run public key encryption scheme to get C_1 and the session key material, encrypt M_b as C_2 under the session key and finally compute $C_3 = H(C_1||C_2||R||M)$
3. run $A_2(C_1||C_2||C_3, st)$ and get the output bit b'. Finally, output bit $b = b'$.

As usual, we will need to simulate answers from the random oracle. This is done using a **H-list** and a **G-list**. For a fresh query h to H, i.e. a query not on the **H-list**, the oracle picks a random answer and similarly for G. We also define a plaintext-extractor as follows:

1. on a query $C' = (C'_1||C'_2||C'_3)$ to the decryption oracle, $\mathcal{B}$ first looks at the **H-list**. For any R' of the proper bit-length ℓ_R appearing in a question to H of the form $C'_1||C'_2||R'||M'$ in the list, and such that the corresponding answer is C'_3, the extractor queries the integer $g^{-R'}C'_1 \bmod n$ to the δ-p-subgroup oracle.

2. If the answer of the oracle is positive, the extractor calls G at R' (extending the **G-list** if needed) and obtains the corresponding session key k. Once this is done, it checks whether M' is the decryption of C'_2

3. As soon as it has found a pair (R', M'), as described above the extractor returns M' as the requested plaintext. If none has been found, the query is rejected.

Note that proper bookkeeping allows to query the δ-p-subgroup oracle at most once for each H-query.

Finally, we explain why the simulation is related to inverting the OU scheme. We define a machine $\mathcal{B}$ which is as above, but receives an input C_1 and a session key to encrypt M_b. This machine performs an additional step when execution has ended. Parsing the questions in the **G-list** and the **H-list**, $\mathcal{B}$ looks for a string R of the appropriate length ℓ_R, appearing in either list. For each such string, $\mathcal{B}$ queries the integer $g^{-R}C_1 \bmod n$ to the δ-p-subgroup oracle. As soon as it finds an element such that the answer of the oracle is positive, $\mathcal{B}$ returns R. Note that the running time t' of $\mathcal{B}$, when it is executed with the help of the plaintext extractor is the running time t of $\mathcal{A}$ plus an additional term $\mathcal{O}(2q_H + q_G)$ to process the data on the lists and call the oracles.

We compare the behavior of several games

1. game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle, and the random oracle is perfect

2. game $\mathcal{G}_2$, where the decryption queries are answered by the plaintext-extractor using a perfect p-subgroup oracle until $\mathcal{B}$ is able to output an answer R. Oracles are still assumed perfect and further decryption queries are answered by an actual decryption oracle.

3. game $\mathcal{G}_3$, which is as $\mathcal{G}_2$ but where the random oracles are simulated until $\mathcal{B}$ is able to output an answer R.

4. game $\mathcal{G}_4$, which is played as $\mathcal{G}_3$ but stops (before asking the G or H oracle) as soon as $\mathcal{B}$ is able to output an answer R.

5. game $\mathcal{G}_5$, where the session key is not derived as prescribed by the cryptosystem but is randomly chosen

6. game $\mathcal{G}_6$, where a δ-p-subgroup oracle is used

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary with the real-life setting. As usual, we bound the difference of probabilities between $\mathcal{G}_1$ and $\mathcal{G}_6$.

To go from $\mathcal{G}_1$ to $\mathcal{G}_2$, we have to bound the probability that the plaintext-extractor rejects a correct ciphertext $C' = (C'_1, C'_2, C'_3)$. There are two cases, depending on whether C'_1 and C_1 encrypt the same value or not. We first cover the second case. Let R' be the decryption of C'_1 under the OU scheme. The situation that we wish to avoid can be described as follows

1. the value of G at R' is set at k'
2. let $M' = \mathsf{D}_{\mathsf{k}'}(C'_2)$; the value of H at $(C'_1||C'_2||R'||M')$ is later set at C'_3.

The conditional probability of the second item, once the first happens, is bounded by $\frac{1}{2^{\ell_H}}$.

We turn to the first case, where the decryption of C'_1 under the OU scheme equals the decryption R of C_1. Here, the value of $G(R)$ is implicitly constrained by the assumption that C is a ciphertext of the test message M_b. Fortunately, $\mathcal{B}$ has already found the OU decryption of C_1 in this case and, therefore, the plaintext-extractor is not called.

Altogether, we have bounded the probability that the plaintext-extractor is wrong by $\frac{q_D}{2^{\ell_H}}$. Discarding the corresponding executions of games $\mathcal{G}_1$ and $\mathcal{G}_2$ leads to perfect simulation. Thus the probabilities that $\mathcal{G}_1$ and $\mathcal{G}_2$ output one differ by at most $\frac{q_D}{2^{\ell_H}}$.

The simulation in game $\mathcal{G}_3$ is perfect since it cannot conflict with the implicit constraint coming from the assumption that C is a ciphertext of the test message M_b. Indeed, R cannot be part of a question to the simulated versions of G or H, where R is, as defined above, the decryption of C_1 under the OU scheme. This is because $\mathcal{B}$ has earlier found the OU decryption of C_1.

To go from $\mathcal{G}_3$ to $\mathcal{G}_4$, we just have to exclude the event of probability $\varepsilon' = \mathsf{Succ}^{\mathsf{ow}}(\mathcal{B})$ that $\mathcal{B}$ has correctly decrypted C_1 before it stops. This bounds the difference of the probabilities that each game outputs one.

The simulation in game $\mathcal{G}_5$ is perfect unless it conflicts with the implicit constraint on G coming from the assumption that C is a ciphertext of a test message M_b. Let R be the decryption of C_1 under the OU scheme. The conflict can only occur if R is of the appropriate length ℓ_R and is queried to G. Since $\mathcal{G}_3$ has aborted beforehand, this cannot happen.

To go from $\mathcal{G}_5$ to $\mathcal{G}_6$, one just needs to estimate the probability that the δ-p-oracle returns a wrong answer. Since it is queried $(2q_H + q_G)$ times, this is bounded by $(2q_H + q_G) \cdot \delta$

Finally, we have bounded the difference between the respective probabilities that $\mathcal{G}_1$ and $\mathcal{G}_6$ output one by:

$$\varepsilon' + \frac{q_D}{2^{\ell_H}} + (2q_H + q_G) \cdot \delta$$

To conclude, consider the probability that $\mathcal{G}_6$ outputs one. In this game, a random string is drawn as a session key and used to encrypt a randomly chosen test message M_b under this key. The adversary outputs one if it has correctly guessed bit b. This is exactly the situation of a semantic distinguisher as defined in section 5.2.1. Therefore, the advantage of the adversary in game $\mathcal{G}_6$ is bounded by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$.

6.4 Summary of the security analysis

Piecing together the results of theorems 8 and 4, we see that EPOC-3 is semantically secure against CCA adversaries, based on the hardness of factoring, if it is derived from the version of the Okamoto-Uchiyama cryptosystem described in [28]. The sequence of reductions under this result can be made tight by an appropriate choice of the parameters

There are technical difficulties to cover the general case, where h and g^p do not necessarily lie in the same subgroup. This comes from the proof of theorem 4, whose proof uses the fact that $g^x \bmod n$ essentially generates the same distribution as the cryptosytem. This is not true in general: for suitable choices of g and h, the quadratic residues modulo n appear with probability $1/2$ in the first case and $1/4$ in the second. One could use $g^x h^r \bmod n$ in place of $g^x \bmod n$ but the analysis remains more subtle than expected: we see how to extend of theorem 4 when $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are mutually prime, with an extra multiplicative loss $1/2$ in the probabilities, but have not attempted to cover the general case.

Similarly, the hypotheses of the proof theorem 4 requires that ℓ_R is $k-1$. A few bits can be discarded with a multiplicative loss $1/2$ in the probabilities, for each bit. However, the submission only requires that $\ell_R \leq k-1$ and this leaves open the possibility that $\ell_R << k-1$, in which case the security proof collapses.

Similarly, ℓ should be larger than $2k$. However, the submission does not include such requirement and this leaves open the possibility that $\ell << 2k$, in which case the security proof collapses.

Finally, the order d of h should be large.

Thus, as for EPOC-1 and EPOC-2, the submission does not make completely clear the role of the various parameters: again, the analysis shows that

1. the bit-length ℓ_R of the random string should be close enough to $k-1$ (the submission only requires $\ell_R \leq k-1$)

2. the bit-length ℓ of the random string r should be larger than $2k$

3. the order of h should be large

However, no part of the submission specifically points out these requirements.

7 Conclusions

Based on our analysis, we think that the version of the p-subgroup assumption, which appears in the submission, is not correct. Since it relies on the assumption, we would not recommend EPOC-1 as it is. We believe that EPOC-2, EPOC-3 are secure, with the proposed parameters. However, based on the submission, we have the following restrictions:

- The specification contains ambiguities and omits additional requirements on the parameters that appear necessary to complete the various security proofs.
- The lower choice of suggested parameters guarantees security for a foreseeable period of time which is rather limited. There are no indications of how to increase the various parameters consistently.
- Although additional requirements on the parameters allow to prove the security of both schemes against adaptive chosen-ciphertext attacks, we note that such proof would not give any conclusive evidence for EPOC-2, when interpreted with the proposed parameters.
- The Gap higher residuosity problem on which EPOC-3 is based appears to have been introduced too recently to form the basis of a cryptosystem.

References

[1] M. Abdalla, M. Bellare and P. Rogaway. DHAES: An encryption scheme based on the Diffie-hellman problem, `http://www-cse.ucsd.edu/users/mihir`

[2] D. Atkins, M. Graff, A. K. Lenstra and P. Leyland, The magic words are squeeming ossifrage, Asiacrypt'94, Lecture Notes in Computer Science 917, (1995), 263–277.

[3] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.

[4] M. Bellare and P. Rogaway. Random Oracles Are Practical: a Paradigm for Designing Efficient Protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.

[5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.

[6] D. Boneh, G. Durfee, and N. Howgrave-Graham, Factoring $N = p^r q$ for large r, Crypto'99, Lecture Notes in Computer Science 1666, (1999), 326–337.

[7] R. P. Brent, Some Parallel Algorithms for Integer Factorisation, Euro-Par 99, Lecture Notes in Computer Science 1685, (1999), 1–22.

[8] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. C. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, P. Zimmermann, Factorization of a 512-Bit RSA Modulus. Eurocrypt'2000, Lecture Notes in Computer Science 1807,(2000), 1–18.

[9] D. Coppersmith, Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known, Eurocrypt'96, Lecture Notes in Computer Science 1070, (1996), 178–189.

[10] R.-M. Elkenbracht-Huizing, An implementation of the number field sieve, it Exp. Math. 5, (1996), 231-253.

[11] Specification of EPOC: Efficient probabilistic public-key encryption.

[12] Self evaluation of EPOC: Efficient probabilistic public-key encryption.

[13] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Crypto'98, Lecture Notes in Computer Science 1462, (1998), 13–25.

[14] D. Dolev, C. Dwork, and M. Naor. Non-Malleable Cryptography. In *Proc. of the 23rd STOC*. ACM Press, New York, 1991.

[15] T. El Gamal, A public key crtyptosystem and signature scheme based on discrete logarithms, *IEEE Trans. on Inform. theory*, 31 (1985), 469–472.

[16] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999.

[17] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.

[18] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.

[19] S. Goldwasser and S. Micali, Probabilistic encryption, *Journal of Computer and System Science* 28, (1984), 270–299.

[20] R. Impagliazzo and D. Zuckermann, How to rectcle random bits, *30th annual symposium on foundations of computer science*, (1989), 248–253.

[21] A. K. Lenstra, H. W. Lenstra and L. Lovász, Factoring polynomials with rational coefficients, *Mathematische Ann.*, 261, (1982), 513–534.

[22] A.K. Lenstra and E. Verheul, Selecting cryptographic key sizes, PKC'2000, Lecture Notes in Computer Science 1751, (2000), 446–465.

[23] N. Lygeros, M. Mizony, P. Zimmermann, A new ECM record with 54 digits, `http://www.desargues.univ-lyon1.fr/home/lygeros/Mensa/ecm54.html`

[24] P. L. Montgomery, A block Lanczos algorithm for finding dependencies over GF(2), Eurocrypt'95, Lecture Notes in Computer Science 921, (1995) 106–120.

[25] M. Naor, O. Reingold, Number-theoretic Constructions of Efficient Pseudo-random Functions, *38-th annual symposium on foundations of computer science*, (1997), 458–467.

[26] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *RSA '2001*, LNCS. Springer-Verlag, Berlin, 2001.

[27] T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC '2001*, LNCS. Springer-Verlag, Berlin, 2001.

[28] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring, Eurocrypt'98, Lecture Notes in Computer Science 1403, (1998), 308–318.

[29] R. Peralta and E. Okamoto, Faster Factoring of Integers of a Special Form , IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences, v. E79-A, n.4 (1996), 489–493.

[30] J. Pollard, Monte Carlo methods for index computation mod p, *Mathematics of Computation*, 32, (1978), 918–924.

[31] RSA Laboratories, Information on the RSA challenge, `http://www.rsa.com/rsalabs/html/challenges/html`

[32] V. Shoup and T. Schweinberger, ACE Encrypt: The Advanced Cryptographic Engine' public key encryption scheme, Manuscript, March 2000. Revised, August 14, 2000.