Evaluation Report on the **HIME-1 Cryptosystem**

1 Introduction

This document is an evaluation of the **HIME-1 Cryptosystem**. Our work is based on the analysis of documents [13, 14, 15]. The present report is organized as follows: firstly, we briefly review the cryptosytem; 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

HIME-1 is based on the hardness of factoring integers n of the form $p^d q$, where p and q are prime numbers - approximately of the same size - such that $p = 3 \bmod 4$, $q = 3 \bmod 4$, and where d is a small odd integer > 1, typically $d = 3$ or 5. Let k be the size in bits of integer pq and let $J(x)$ denote the Jacobi symbol of x with respect to n. The basic functions f, g on which HIME-1 is based are defined by

$$\begin{array}{rcl} f : \{0,1\}^{k-2} & \longrightarrow & \mathbb{Z}_n^\star \\ x & \longmapsto & x^{2n} \bmod n \end{array} \qquad \begin{array}{rcl} g : \{0,1\}^{k-2} & \longrightarrow & \mathbb{Z}_n^\star \times \{-1,1\} \\ x & \longmapsto & (f(x), J(x)) \end{array}$$

We have the following:

Theorem 1 *The function g is one-to-one.*

Proof: Note that, since d is odd and p, q are congruent to 3 modulo 4, $J(-1) = 1$. Let $y = f(x)$. We have

$$x^{2n} = y \bmod p$$

Since n is prime to $p-1$, this allows to compute $\nu = n^{-1} \bmod (p-1) = q^{-1} \bmod (p-1)$, from which we obtain

$$x^2 = y^\nu \bmod p$$

This equation has two square roots. As is well-known, one is computed by $x_p = (y^\nu)^{\frac{p+1}{4}}$ and the other by $-x_p \bmod p$. A similar argument, with ν replaced by $\nu = n^{-1} \bmod (q-1) = p^{-d} \bmod (q-1)$, yields two values x_q and $-x_q \bmod q$, such that $x \bmod q$ is one of these values. Using Chinese remaindering, we come out with a choice of four possible values for $x \bmod pq$, two of them with Jacobi symbol 1 and the two other with Jacobi symbol -1. From $J(x)$, we can discard two values and there remain a pair $\pm x \bmod pq$. Since x is $< 2^{k-2}$, we can select the correct value without any ambiguity.

Remark: Note that we had to assume that n is prime with $p-1$ and $q-1$. This is implicit in [13] but follows from the key generation algorithm (section **3.2.1**), which guarantees that p and q have exactly the same bit length.

Before explaining how the HIME-1 cryptosystem applies the above transformation, it is useful to 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 $r \in \Omega$
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ outputs the plaintext m associated to the ciphertext c.

The key generation algorithm $\mathcal{K}(1^k)$ of the HIME-1 Cryptosystem produces $n = p^d q$, where p, q are two odd random primes such that $p = 3 \bmod 4$ and $q = 3 \bmod 4$, $|pq| = k$, and d is an odd integer ≥ 3. The public key pk is the pair (n, k) and thus defines the above function g. The secret key sk is the pair (p, q), which helps inverting g. The message space is $\mathcal{M} = \{0,1\}^{k-k_0-k_1-2}$, where k_0, k_1, are appropriate parameters. The encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ uses two hash functions G and H:

$$G : \{0,1\}^{k_0} \longrightarrow \{0,1\}^{k-k_0-2} \text{ and } H : \{0,1\}^{k-k_0-2} \longrightarrow \{0,1\}^{k_0}$$

It takes $m \in \mathcal{M}$ and a random value $r \xleftarrow{R} \{0,1\}^{k_0}$, and computes a $k-2$-bit integer $x = a \| b$, by OAEP formatting (see [3]), as follows:

$$a = m0^{k_1} \oplus G(r) \text{ and } b = r \oplus H(a)$$

where $\oplus$ is bitwise modulo 2 addition. The ciphertext is $c = g(x)$, with $x = a \| b \in \{0,1\}^{k-2}$. The decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ extracts $x = a \| b$ and computes:

$$r = b \oplus H(a) \text{ and } M = a \oplus G(r)$$

It checks that M is correctly formatted, with its k_1 trailing bits zero and returns the $k - k_0 - k_1 - 2$ leading bits. Otherwise it returns "Reject".

We refer to [13] for a precise definition of G and H and for an accurate choice of the parameters.

2.2 Comments on the specification

Document [13] needs further editing. There are ambiguities. For example, notation q^{-1} on page 7, line 11, is unclear: it should be stated that the inverse is taken modulo $p-1$. Also, the key generation algorithm (section **2.2.1** of document [13]) is vague: it is explained that prime numbers p, q with 256 bits are generated but nothing guarantees that p^3q is actually 1024 bit long (it could be slightly less). There are errors. For example, on page 10, line 9, it is stated that message m is such that $0 < m < 2^{k-2}$, whereas $k-2$ should be replaced by $k - k_0 - k_1 - 2$. Most ambiguities and errors can be corrected in a straightforward manner. However, some are more serious. Exception handlings in the decryption operation are simply not addressed. On line 7 of the decryption pseudo-code in section 3.2.3 of document [13], values m' such that $m' \geq 2^{k-2}$ are replaced by $pq - m'$. However, if incorrect ciphertexts are submitted, it is very possible to come out with a situation where m' remains $\geq 2^{k-2}$ even after the execution of instruction 7. Such values of m' should not be submitted to the $convert^{-1}$ operation. Still, no mention of this case is made. Similarly, neither the $convert^{-1}$ nor the decryption pseudo-code calling this function checks that the redundancy is correct. It is absolutely necessary to verify that the string w in section **3.2.5** consists of zeros. Such mistakes are particularly embarrassing as they pollute the subtle security arguments that have to be provided in favour of the cryptosystem.

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

On page 6 of document [13], it is claimed that

> the encryption function of the basic scheme is a deterministic trapdoor permutation.

and, in theorem 1.1 of document [14], its is stated that, if d is small enough,

the basic scheme is OW-CPA under the assumption of the difficulty of the factoring problem.

These statements needs some clarification for reasons that we now develop.

The mathematical definition of a permutation applies to functions whose inputs and outputs have the same range: for example the RSA permutation operates on integers smaller than the modulus. Here, the basic function is certainly one-to-one, as shown in section 2.1, but its output range is never specified in documents [13, 14]. In any case, outputs are $|n|$ bit long, whereas inputs are k bit long. Thus, the wording *permutation* is improper.

Even though the word permutation is misleading, the proof of theorem 1.1 of document [14], quoted above, is correct. It is based on the observation that it is possible to extend the basic function, using exactly the same formula. The extended function is not one-to-one but takes x and $pq - x$ to the same value as shown by Lemma **1.1** of document [13]. The proof is not complete in the submission and we include our version here:

Lemma 1 *Let $n = p^d q$ with prime numbers p, q, and $d < \min\{p, q\}$. Let F be a function on $\mathbb{Z}_n$ (the extension of f on $\mathbb{Z}_n$) such that $F(x) = x^{2n} \bmod n$. Then $F(x) = F(-x + pq)$.*

proof: We first explain the condition $d < \min\{p, q\}$: for $i < d$, $i!$ is prime relatively to p and q, whereas $2n \times \cdots \times 2n - i + 1$ is divisible by n, and thus by $p^d q$. Therefore, $\binom{2n}{i}$ is divisible by n. We then get:

$$\begin{aligned}
F(-x+pq) &= \sum_{i=0}^{i=2n} \binom{2n}{i} (-x)^{2n-i} (pq)^i \\
&= x^{2n} + \sum_{i=1}^{i=d-1} \binom{2n}{i} (-x)^{2n-i} (pq)^i + (pq)^d \sum_{i=d}^{i=2n} \binom{2n}{i} (-x)^{2n-i} (pq)^{i-d} \\
&= x^{2n} + n \times \sum_{i=1}^{i=d-1} \left(\binom{2n}{i} \times \frac{1}{n} \right) \times (-x)^{2n-i} (pq)^i \\
&\quad + n \times q^{d-1} \sum_{i=d}^{i=2n} \binom{2n}{i} (-x)^{2n-i} (pq)^{i-d} = x^{2n} \bmod n = F(x)
\end{aligned}$$

This finishes the proof.

From the lemma, we see that, applying any inversion algorithm to the image by the extended function of an input x such that $pq - 2^{k-2} < x < pq$, produces $pq - x$ and therefore factors n. This happens with significant probability $\geq 1/4$. More formally:

Theorem 2 *If d is small enough ($d < \min\{p, q\}$), the function g is* one-way*, based on the factorization of n: if a machine $\mathcal{A}$ is able to invert g with probability ε, within time bound t, there exists a machine $\mathcal{B}$ that is able to factor n with probability $\varepsilon' \geq \varepsilon/4$, within time bound $t' \leq t+\tau$, where the overhead τ accounts for performing computations with integers of size $|n|$ and is bounded by $\mathcal{O}((\frac{(d+1)k}{2})^3)$.*

Proof: Let us consider an adversary $\mathcal{A}$ able to break the one-wayness of g with probability ε, within time bound t:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[x \stackrel{R}{\leftarrow} \{0,1\}^{k-2}, y \leftarrow g(x), z \leftarrow \mathcal{A}(y) : z \in \{0,1\}^{k-2} \wedge g(z) = y] > \varepsilon$$

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

As above, we denote by G the extension of g: for any $x \in \mathbb{Z}_n$, $G(x) \leftarrow (F(x), J(x))$. Observe that $G(x) = G(pq - x)$, for any $x \in \mathbb{Z}_n$, since $J(pq - x) = J(x)$:

$$\begin{aligned} J(pq-x) &= \left(\frac{pq-x}{n}\right) = \left(\frac{pq-x}{p}\right)^d \times \left(\frac{pq-x}{q}\right) \\ &= \left(\frac{-x}{p}\right)^d \times \left(\frac{-x}{q}\right) = \left(\frac{x}{p}\right)^d \times \left(\frac{x}{q}\right) = \left(\frac{x}{n}\right) = J(x) \end{aligned}$$

since both $p = 3 \bmod 4$ and $q = 3 \bmod 4$, which implies $\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = 1$.

We use the above adversary $\mathcal{A}$ to factor n, by describing a polynomial Turing Machine $\mathcal{B}$ that makes calls to $\mathcal{A}$:

1. $\mathcal{B}$ chooses $x \stackrel{R}{\leftarrow} \{0,1\}^k$;
2. $\mathcal{B}$ computes $y \leftarrow G(x)$;
3. $\mathcal{B}$ runs $\mathcal{A}$ on y, and gets z;
4. $\mathcal{B}$ returns $(n/\gcd(n, x+z))^{1/(d-1)}$.

We study the probability ν that $\mathcal{B}$ returns factor p of n:

$$\nu = \Pr[p \leftarrow \mathcal{B}] = \Pr[x \stackrel{R}{\leftarrow} \{0,1\}^k, y \leftarrow G(x), z \leftarrow \mathcal{A}(y) : z = pq - x]$$

Let us define $X_0 = [0, 2^{k-2}[$ and $X_1 =]pq - 2^{k-2}, pq]$. Both X_0 and X_1 have the same size and are included in $\{0,1\}^k$. Furthermore

$$\begin{aligned} \mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) &= \Pr[x \stackrel{R}{\leftarrow} X_0, y \leftarrow G(x), z \leftarrow \mathcal{A}(y) : z \in X_0 \wedge G(z) = y] \\ &= \Pr[x \stackrel{R}{\leftarrow} X_0, y \leftarrow G(pq - x), z \leftarrow \mathcal{A}(y) : z \in X_0 \wedge G(z) = y] \\ &= \Pr[x \stackrel{R}{\leftarrow} X_1, y \leftarrow G(x), z \leftarrow \mathcal{A}(y) : z \in X_0 \wedge G(z) = y] \end{aligned}$$

Therefore, with the probability distribution $\{x \stackrel{R}{\leftarrow} \{0,1\}^k, y \leftarrow x^{2n} \bmod n, z \leftarrow \mathcal{A}(y)\}$,

$$\begin{aligned} \nu &\geq \Pr[G(z) = y \wedge (x \in X_1) \wedge (z \in X_0)] \\ &\geq \Pr[G(z) = y \wedge (z \in X_0) \mid (x \in X_1)] \times \Pr[x \in X_1] \\ &\geq \mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) \times \Pr[x \in X_1] \end{aligned}$$

On the other hand,

$$\Pr[x \in X_1] \geq \frac{2^{k-2}}{2^k} = \frac{1}{4}$$

This completes the proof of the theorem.

What is crucial in the above is the ability to obtain the set of images of the basic function as a subset of the image of the open interval $(0, 2^k)$ by some function g computed by an efficient algorithm.

3.2 Size of the parameters

As was just observed the security of the basic scheme is essentially equivalent to the hardness of factoring integers n of the form $n = p^d q$, even if there is a small security loss in terms of exact security.

Thus, the security of the basic scheme is basically equivalent to the hardness of factoring integers n of the form $n = p^d q$. 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.

Before entering into a more precise discussion, let us mention that the idea is of having moduli of the form $p^d q$ is not new. For example, it appears in [23]. This remark is not in terms of intellectual property but rather in terms of the novelty of the idea.

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 of 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 adequately addressed by the requirement that each of $p-1$, $q-1$ has a large prime factor (document [13], page 10).

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 with 54 digits of a 127-digit composite number n was found with GMP-ECM, a free implementation of the Elliptic Curve Method (see [17]). The limit used was $B = 15,000,000$.

In a recent paper [5], 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. It can be noted that, if Brent's prediction is correct, parameters of the scheme under review will become insecure at year $Y = 2014$.

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 one gets a 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 [21]). It was factored in april 1994 and took approximately 5000 mips-years (see [1]).

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 [11]). 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 ([7]). 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 [18]). In [7], 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 [5] and produces $Y = 2018$ for $D = 309$, i.e. for a 1024 bit modulus, as

proposed in HIME-1.

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

In recent work (see [4]), 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 [16]. 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 $\sqrt{\log p}$. For the suggested parameters of HIME-1, which set $d = 3$, the algorithm is certainly impractical.

3.2.4 Conclusion

Based on current estimates, it appears that the proposed parameters for HIME-1 should remain secure for at least thirteen years. Surprisingly, the use of moduli with smaller factors makes ECM factoring the main threat to the cryptosystem. Although predictions should be taken with great care, it seems that this shortens the "lifetime" of the proposed parameters by something like four years. One might even argue for a larger estimate since, contrary to NFS, ECM factoring does not face the bottleneck of linear algebra and, accordingly, predictions might be more accurate. Furthermore, we believe that the behavior of ECM on integers of the form $p^3 q$ has not been documented through experiments. There are indications that it is possible to slightly speed up ECM for numbers of the form $p^2 q$ (see [19]). Even though the improvement does not apply to $n = p^d q$, with odd d, it is good cryptographic practice to avoid introducing additional structure and non generic objects. Thus, having p appearing three times in the prime decomposition of n is questionable.

4 Security Analysis

The self-evaluation report [14] does not includes a serious security analysis. It simply refers to [3]. However, as observed in section 3.1, it improperly claims that the basic function g is a permutation, a property that is required to apply results from [3]. Thus, we have found necessary to undertake our own security analysis. This appears absolutely needed in view of the recent flaw discovered in the analysis of OAEP (see [22, 12]). We have to check whether or not the security analysis can be performed for HIME-1.

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

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = 2 \times \Pr_{\substack{b \stackrel{R}{\leftarrow} \{0,1\} \\ r \stackrel{R}{\leftarrow} \Omega}} \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{K}(1^k), (m_0, m_1, s) \leftarrow A_1(\mathsf{pk}) \\ c \leftarrow \mathcal{E}_{\mathsf{pk}}(m_b; r) : A_2(c, s) \stackrel{?}{=} b \end{array} \right] - 1$$

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, called *non-malleability* [10]. 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 [2] that, under CCA, semantic security and non-malleability are equivalent. This is the strongest security notion currently considered.

We turn to the security analysis. We want to prove that the HIME-1 scheme is IND-CCA in the random oracle model, based on the assumption that factoring is hard. More precisely, we wish to turn a CCA adversary $\mathcal{A}$ into a machine inverting g and apply theorem 2. Unfortunately, as shown in [22], this is hopeless. In place, we will turn $\mathcal{A}$ into a machine that *partially inverts* g. By this, we mean that it outputs a list

of bit-strings, including, with significant probability, the $k - k_0 - 2$ leading bits of the targeted preimage. We thus establish the following exact security result.

Theorem 3 *Let $\mathcal{A}$ be a CCA–adversary $\mathcal{A}$ attacking $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, 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 $\mathcal{B}$ partially inverting g, with success probability ε' and within time bound t' where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{2} \times \left(1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-2}}\right) - \frac{q_G}{2^{k-3}} - \frac{q_D}{2^{k_1-1}} - \frac{(2q_G+1)\cdot q_D}{2^{k_0}} \\ t' &\leq t + q_G \cdot q_H \cdot \tau \end{aligned}$$

where τ is the time needed to execute the encryption algorithm and is bounded by $\mathcal{O}\left(\left(\frac{(d+1)k}{2}\right)^3\right)$

To prove the above, we follow [3]: we first show the semantic security against chosen-plaintext attacks (IND-CPA), and next we prove the existence of a plaintext-extractor.

4.2 Semantic security

4.2.1 Description of the simulator

We describe a simulator $\mathcal{B}$ which tentatively provides the adversary with the same view as in a real attack. Let $\mathcal{A} = (A_1, A_2)$ be an adversary against the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, under a chosen-plaintext attack. Within time bound t, $\mathcal{A}$ asks q_G and q_H queries to the hash functions G and H respectively, and distinguishes the correct plaintext with an advantage greater than ε. Let us describe the simulator $\mathcal{B}$:

1. $\mathcal{B}$ first runs $\mathcal{K}(1^k)$ to obtain the function g, defined from the public key
2. then $\mathcal{B}$ is given $y \leftarrow g(x)$, for $x \stackrel{R}{\leftarrow} \{0,1\}^{k-2}$, for which it wants to find a preimage; since g is one-to-one, the preimage is x, which we split as $x = \alpha \| \beta$
3. 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 a random bit b, and then defines $C \leftarrow y$, to be a ciphertext of m_b
4. $\mathcal{B}$ runs $A_2(C, s)$ and finally gets an answer b'. Finally, $\mathcal{B}$ outputs a preimage of y, if one has been found from the queries asked to G and H (see below), or Fail

Of course, during the entire simulation, $\mathcal{B}$ also has to simulate answers from the random oracle. This is done as follows:

- For a fresh query γ to G, build $z = \delta \| (\gamma \oplus H_\delta)$ for all previously asked queries δ to H with answer H_δ, and check whether $y = g(z)$. If for some δ this relation holds, then g has been inverted, and one can still correctly simulate G, by answering $G_\gamma = \delta \oplus m_b 0^{k_1}$. This is indeed a uniformly distributed value: since g is a one-to-one, we get $z = x$ and thus $\delta = \alpha$, which is uniformly distributed, by definition of x. Otherwise, output a random value G_γ.

- For a fresh query δ to H, output a random value H_δ. Additionally, for any query γ previously asked to G with answer G_γ, build $z = \delta \| (\gamma \oplus H_\delta)$, and check whether $y = g(z)$. If for some γ this relation holds, then g has been inverted.

Once a preimage of y has been found, one could simply output it and stop the game. But for the analysis, we assume the game goes on and that $\mathcal{B}$ only outputs the answer (or Fail, if no preimage has been found) after A_2 has answered b'.

The simulation is perfect from the random oracle point of view, since, as was seen, a new uniformly distributed value is returned for each new query. Still, it may be imperfect, due to the implicit constraint coming from the assumption that C is a ciphertext of m_b; the corresponding random tape r is such that:

$$r \leftarrow H(\alpha) \oplus \beta \text{ and } G(r) \leftarrow \alpha \oplus m_b 0^{k_1}$$

Since $H(\alpha)$ is randomly defined, r can be seen as a random variable. Let us denote by AskG the event that the query r has been asked to G, and by AskH the event that query α has been asked to H, and by FAskH the event that query α has been asked to H during the find-stage (by A_1). Let us also denote

- by FBad the event that r has been queried to G in the find-stage (by A_1), but answered by a value different from $\alpha \oplus m_0 0^{k_1}$ or $\alpha \oplus m_1 0^{k_1}$

- and by GBad the event that query r has been asked to G in the guess-stage (by A_2), but answered by a value different from $\alpha \oplus m_0 0^{k_1}$ or $\alpha \oplus m_1 0^{k_1}$. Note that, if this happens, $H(\alpha)$ has not been asked previously since the control during the G-simulation would entail $G_r \leftarrow \delta \oplus m_b 0^{k_1} = \alpha \oplus m_b 0^{k_1}$

One may observe that each event, FBad or GBad, implies AskG. As explained above, FBad and GBad are the only events which make the simulation imperfect. we set:

$$\mathsf{Bad} = \mathsf{FBad} \vee \mathsf{GBad}$$

4.2.2 Probabilistic analysis

We denote by $\Pr[\cdot]$ the probabilities in the real attack, and by $\mathsf{pr}[\cdot]$ the probabilities in the simulated game.

Note that the adversary cannot gain any advantage, in the real game, without having asked r to G. Indeed the simulation is otherwise perfect, since $\neg\mathsf{AskG}$ implies $\neg(\mathsf{FBad} \vee \mathsf{GBad})$, and it is clearly independent of b. Thus the probability of correctly guessing b not having asked r is exactly one half. From this, we bound $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})$ by:

$$\begin{aligned}
\varepsilon &= 2 \times \Pr[A = b \,|\, \mathsf{AskG} \wedge \mathsf{AskH}] \times \Pr[\mathsf{AskG} \wedge \mathsf{AskH}] \\
&\quad +2 \times \Pr[A = b \,|\, \mathsf{AskG} \wedge \neg\mathsf{AskH}] \times \Pr[\mathsf{AskG} \wedge \neg\mathsf{AskH}] \\
&\quad +2 \times \Pr[A = b \,|\, \neg\mathsf{AskG}] \times \Pr[\neg\mathsf{AskG}] - 1 \\
&\leq 2 \times \Pr[\mathsf{AskG} \wedge \mathsf{AskH}] + 2 \times \Pr[\mathsf{AskG} \wedge \neg\mathsf{AskH}] + (2 \times \Pr[A = b \,|\, \neg\mathsf{AskG}] - 1) \\
&\leq 2 \times \Pr[\mathsf{AskG} \wedge \mathsf{AskH}] + 2 \times \Pr[\mathsf{AskG} \wedge \neg\mathsf{AskH}] + 0.
\end{aligned}$$

In the following, we are interested into the probabilities for the simulated game, which is perfect unless FBad or GBad happens. This yields:

$$\varepsilon \leq 2 \times (\mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \,|\, \neg\mathsf{Bad}] + \mathsf{pr}[(\mathsf{AskG} \wedge \neg\mathsf{AskH}) \,|\, \neg\mathsf{Bad}]) \,.$$

Remark that

$$(\mathsf{AskG} \wedge \neg\mathsf{AskH}) \wedge \neg\mathsf{Bad} = (\mathsf{AskG} \wedge \neg\mathsf{AskH}) \wedge \neg(\mathsf{FBad} \wedge \mathsf{GBad})$$

is exactly the conjunction of the events that r has been asked to G, that α has not been previously asked to H and (coming from $\neg(\mathsf{FBad} \wedge \mathsf{GBad})$), that the answer has been either $\alpha \oplus m_0 0^{k_1}$ or $\alpha \oplus m_1 0^{k_1}$. This probability is less than

$$q_G \cdot 2^{-k_0} \times 2 \cdot 2^{-k+k_0+2} = 2q_G \cdot 2^{-k+2} = q_G \cdot 2^{-k+3}$$

Therefore,

$$\mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \,|\, \neg\mathsf{Bad}] \geq \varepsilon/2 - q_G \cdot 2^{-k+3}/\mathsf{pr}[\neg\mathsf{Bad}]$$

and thus,

$$\begin{aligned}
\mathsf{pr}[\mathsf{AskG} \wedge \mathsf{AskH}] &\geq \mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \wedge \neg\mathsf{Bad}] \geq \mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \,|\, \neg\mathsf{Bad}] \times \mathsf{pr}[\neg\mathsf{Bad}] \\
&\geq \varepsilon/2 \times \mathsf{pr}[\neg\mathsf{Bad}] - q_G \cdot 2^{-k+3}
\end{aligned}$$

To conclude, we just have to evaluate the probability $\mathsf{pr}[\mathsf{Bad}] = \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad}]$:

$$\begin{aligned}
\mathsf{pr}[\mathsf{Bad}] &= \mathsf{pr}[\mathsf{Bad} \,|\, \neg\mathsf{FAskH}] \times \mathsf{pr}[\neg\mathsf{FAskH}] + \mathsf{pr}[\mathsf{Bad} \,|\, \mathsf{FAskH}] \times \mathsf{pr}[\mathsf{FAskH}] \\
&\leq \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \,|\, \neg\mathsf{FAskH}] + \mathsf{pr}[\mathsf{FAskH}]
\end{aligned}$$

Firstly, the randomness of α, which is uniformly distributed in $\{0,1\}^{k-k_0-2}$, implies that it is asked to H in the first stage with probability less than $q_H/2^{k-k_0-2}$: $\mathsf{pr}[\mathsf{FAskH}] \leq q_H \cdot 2^{-k+k_0+2}$.

Secondly, one observes that the conditional events FBad or GBad, knowing that ¬FAskH holds, correspond to a situation where $\mathcal{A}$ asks r to G without having asked α to H yet. When queried, the random variable r is undefined. Thus FBad or GBad become true if, later, $H(\alpha)$ is set to a value v such that $v \oplus \beta$ has been asked to G: $\mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\mathsf{FAskH}] \leq q_G \cdot 2^{-k_0}$. Thus,

$$\mathsf{pr}[\mathsf{Bad}] \leq q_G \cdot 2^{-k_0} + q_H \cdot 2^{-k+k_0+2}$$

Finally, the probability that $\mathcal{B}$ outputs x is greater than

$$\varepsilon/2 \times (1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0+2}) - q_G \cdot 2^{-k+3}$$

4.3 Plaintext–Extractor.

If one wants to consider $\mathcal{A}$ as a chosen-ciphertext adversary, $\mathcal{B}$ has to be able to simulate the decryption oracle: on a query c' to the decryption oracle, $\mathcal{B}$ looks at the query/answer list (γ, G_γ) obtained from G and at the query/answer list (δ, H_δ) obtained from H. Then, for each pair of elements coming from both lists, it defines

$$a = \delta, b = \gamma \oplus H_\delta, M = G_\gamma \oplus \delta$$

and checks whether

$$c' = g(a\|b) \text{ and } [M]_{k_1} = 0^{k_1}$$

where $[M]_{k_1}$ denotes the k_1 trailing bits of M. If this happens, then $\mathcal{B}$ outputs the $k - k_0 - k_1 - 2$ leading bits of M, $[M]^{k-k_0-k_1-2}$ as the requested plaintext. Otherwise, "Reject" is returned.

Before performing our analysis, let us check that this simulation uniquely defines a possible plaintext. This comes form the fact that g is one-to-one: the value of a is uniquely defined, and thus, the same is true of δ and H_δ. Similarly, b is uniquely defined, and also γ and G_γ. We conclude that at most one M may be selected, for which either $[M]_{k_1} = 0^{k_1}$ or not.

Given a ciphertext $c' = g(a'\|b')$, we let $r' = H(a') \oplus b'$ and we denote by AskG' the event that query r' has been asked to G, and by AskH' the event that query a' has been asked to H'. The simulation may only fail by rejecting a valid ciphertext. Let us denote by Fail this event:

$$\begin{aligned}
\mathsf{pr}[\mathsf{Fail}] &= \mathsf{pr}[\mathsf{Fail} \wedge \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskG}' \wedge \neg\mathsf{AskH}'] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskG}' \wedge \mathsf{AskH}'] \\
&= \mathsf{pr}[\mathsf{Fail} \wedge \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskG}' \wedge \neg\mathsf{AskH}'] \\
&\leq \mathsf{pr}[\mathsf{Fail} \mid \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{AskG}' \wedge \neg\mathsf{AskH}']
\end{aligned}$$

Following [3], we would wish to argue that, if r' has not been asked to G, the probability that $[a' \oplus G(r')]_{k_1} = 0^{k_1}$ is less than 2^{-k_1}. On the other hand, the probability

to have asked $G(r')$, without having asked $H(a')$ which should leave r' random, is less than $q_G \cdot 2^{-k_0}$. Granted this, one could conclude that

$$\Pr[\mathsf{Fail}] \leq 2^{-k_1} + q_G \cdot 2^{-k_0}.$$

Unfortunately, the above argument is not correct [22], when the decryption oracle interacts with the adversary. This is because, as noticed in the previous section, there is an implicit constraint coming from the assumption that C is a ciphertext of m_b; the corresponding random tape ρ is such that:

$$\rho \leftarrow H(\alpha) \oplus \beta \text{ and } G(\rho) \leftarrow \alpha \oplus m_b 0^{k_1}$$

Now, if it turns out that $r' = \rho$, then one cannot claim anything on the probability that $[a \oplus G(r')]_{k_1} = 0^{k_1}$, even if r' has not been asked from G. Denote by RBad the event that $r' = \rho$. Similarly, let ABad be the event that $a' = \alpha$.

We will show that, without AskH, both events are rare (see [12]). Then we argue that, discarding them, makes the failure probability small.

$$\begin{aligned} & \mathsf{pr}[\mathsf{Fail} \wedge (\mathsf{RBad} \vee \mathsf{ABad}) \mid \neg\mathsf{AskH}] \\ = \; & \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{ABad} \mid \neg\mathsf{AskH}] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{RBad} \wedge \neg\mathsf{ABad} \mid \neg\mathsf{AskH}] \\ \leq \; & \mathsf{pr}[\mathsf{Fail} \mid \mathsf{ABad} \wedge \neg\mathsf{AskH}] + \mathsf{pr}[\mathsf{RBad} \mid \neg\mathsf{ABad} \wedge \neg\mathsf{AskH}] \end{aligned}$$

Within the formula, the second event is interpreted as $r' = \rho$, provided that $a' \neq \alpha$ and $H(\alpha)$ is unknown, and thus unpredictable. Event $r' = \rho$ reads as $H(\alpha) = H(a') \oplus b' \oplus \beta'$, which cannot occur with probability greater than 2^{-k_0}. The first event in the formula can be analyzed, by means of AskG':

$$\begin{aligned} \mathsf{pr}[\mathsf{Fail} \mid \mathsf{ABad} \wedge \neg\mathsf{AskH}] \; = \; & \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskG}' \mid \mathsf{ABad} \wedge \neg\mathsf{AskH}] \\ & + \mathsf{pr}[\mathsf{Fail} \wedge \neg\mathsf{AskG}' \mid \mathsf{ABad} \wedge \neg\mathsf{AskH}] \\ \leq \; & \mathsf{pr}[\mathsf{AskG}' \mid \mathsf{ABad} \wedge \neg\mathsf{AskH}] \\ & + \mathsf{pr}[\mathsf{Fail} \mid \neg\mathsf{AskG}' \wedge \mathsf{ABad} \wedge \neg\mathsf{AskH}] \end{aligned}$$

Once again, the former means that r' is asked to G, whereas $a' = \alpha$ and $H(\alpha)$ remains unpredictable, and thus $H(a')$ and r' as well. Then, the probability that r' has been asked is less than $q_G/2^{k_0}$. The latter means that the redundancy holds, whereas $H(a')$ and $G(r')$ have not been asked. Observe that $a' = \alpha$ implies $r' \neq \rho$ since g is one-to-one, and thus $G(r')$ is unpredictable. Hence, the redundancy cannot hold with probability greater than 2^{-k_1}. Finally,

$$\mathsf{pr}[\mathsf{Fail} \wedge (\mathsf{RBad} \vee \mathsf{ABad}) \mid \neg\mathsf{AskH}] \leq 2^{-k_1} + (q_G + 1) \cdot 2^{-k_0}$$

Using the same argument as in [3], we get:

$$\begin{array}{l}\mathsf{pr}[\mathsf{Fail} \wedge \neg(\mathsf{RBad} \vee \mathsf{ABad}) \wedge \neg(\mathsf{AskG}' \wedge \mathsf{AskH}') \mid \neg\mathsf{AskH}] \\ \mathsf{pr}[\mathsf{Fail} \wedge \neg(\mathsf{RBad} \vee \mathsf{ABad}) \wedge \neg(\mathsf{AskG}' \wedge \mathsf{AskH}')] \leq 2^{-k_1} + q_G \cdot 2^{-k_0}\end{array}$$

It is clear that, granted $\neg(\mathsf{AskG}' \wedge \mathsf{AskH}')$, the above simulator cannot fail. This yields:

$$\mathsf{pr}[\mathsf{Fail} \wedge \neg(\mathsf{RBad} \vee \mathsf{ABad}) \wedge (\mathsf{AskG}' \wedge \mathsf{AskH}')] = 0$$

Accordingly,

$$\mathsf{pr}[\mathsf{Fail} \mid \neg\mathsf{AskH}] \leq 2 \cdot 2^{-k_1} + (2q_G + 1) \cdot 2^{-k_0}$$

Therefore, unless α has been asked to H, all decryption queries are correctly simulated with probability greater than

$$(1 - 2^{-k_1+1} - (2q_G + 1) \cdot 2^{-k_0})^{q_D} \geq 1 - q_D \cdot 2^{-k_1+1} - q_D \cdot (2q_G + 1) \cdot 2^{-k_0}$$

Thus, having given the simulator $\mathcal{B}$ access to the plaintext-extractor, we have either inverted g or else found the bit-string α, consisting of the $k - k_0 - 2$ leading bits of the targeted preimage of g, in the list of queries to H, with probability greater than

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})/2 \times \left(1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0+2}\right) - q_G \cdot 2^{-k+3} - q_D \cdot 2^{-k_1+1} - (2q_G+1) \cdot q_D \cdot 2^{-k_0}$$

This concludes the proof.

4.4 Complexity of the Reduction.

Let us now consider the time complexity of this "reduction", which is the running time of the simulator $\mathcal{B}$. Both the simulator $\mathcal{B}$ and the plaintext-extractor have to look at the query/answer list (γ, G_γ) obtained from G and at the query/answer list (δ, H_δ) obtained from H, and compute for each pair $((\gamma, G_\gamma), (\delta, H_\delta))$,

$$a = \delta, b = \gamma \oplus H_\delta, M = G_\gamma \oplus \delta$$

as well as $g(a\|b)$. Proper bookkeeping allows to do this only once for each pair. Each operation is linear in k, excepted for $g(a\|b)$, whose complexity is denoted by τ, the cost of a modular exponentiation.

Finally, the time complexity of the overall reduction is

$$t + q_G \cdot q_H \cdot (\tau + \mathcal{O}(k)), \text{ with } \tau = \mathcal{O}\left(\left(\frac{(d+1)k}{2}\right)^3\right)$$

4.5 Equivalence with factorization

Combining with the results in section 3.1, we get, in the random oracle model:

Theorem 4 *Let $\mathcal{A}$ be a CCA–adversary attacking $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, 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. Then, there exists an algorithm $\mathcal{B}$ able to extract a prime factor of any integer $n = p^d q$ produced by the key generation algorithm of HIME-1, with success probability ε' and within time bound t' where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{8} \times \left(1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-2}}\right) - \left(\frac{q_G}{2^{k-1}} + \frac{q_D}{4} \times \left(\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}\right)\right) \\ t' &\leq t + (q_G \cdot q_H + 1) \cdot \tau + q_H \times \gamma \end{aligned}$$

where γ is the running time of Coppersmith root finding algorithm from [8].

Combining the reductions in the proofs of theorems 2 and 3, one obtains, from the simulation on $y = x^{2n} \bmod n$, the leading bits a of $z = pq - x$ in a list with q_H elements, with probability greater than

$$\begin{aligned} & \left(\frac{\varepsilon}{2} \times \left(1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-2}}\right) - \frac{q_G}{2^{k-3}} - q_D \times \left(\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}\right)\right) \times \frac{1}{4} \\ \geq\ & \frac{\varepsilon}{8} \times \left(1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0-2}}\right) - \frac{q_G}{2^{k-1}} - \frac{q_D}{4} \times \left(\frac{1}{2^{k_1-1}} + \frac{2q_G+1}{2^{k_0}}\right). \end{aligned}$$

This means that we have candidate approximations $A = a + x = pq - u$ for the unknown value pq, where a comes from the list and where $u \leq 2^{k_0}$. Raising to the d-th power, we obtain a polynomial of degree d, vanishing at $A + u$, modulo n. Coppersmith algorithm [8] can find root u and therefore the factorization of n, provided $|u| < N^{\frac{1}{d}} \simeq 2^{\frac{(d+1)}{2d}}$. When $d = 3$, this means at most $2k/3$ bits and is achieved with the proposed parameters.

4.6 Practical security estimates

We try to understand whether the figures shown above are meaningful for practical parameters, considering the time an adversary could spend on breaking semantic security. We set, as in the specification, $k_0 = 128$ and $k_1 = 126$, $d = 3$ and try to obtain a lower bound on $k \geq 512$. Taking, as many authors, the usual values for q_G, q_H and q_D:

$$q_G \sim q_H \sim 2^{60} \text{ and } q_D \sim 2^{30}$$

we obtain that an adversary could be used for factoring with success probability ε', within a time bound t' where

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{8} \times \left(1 - 2^{-68} - 2^{-322}\right) - \left(2^{-451} + 2^{28} \times \left(2^{-125} + 2^{-67}\right)\right) \approx \frac{\varepsilon}{8} - 2^{-39} \\ t' &\leq t + 8 \times (2^{120} + 1) \cdot k^3 \end{aligned}$$

Therefore, an adversary able to learn one bit with advantage greater than $1/2$, within time less than $2^{116}k^3$ can be used to factor n within less than $2^{123}k^3$.

In the table below, we compare the factoring time of the reduction with those coming from the estimates for the complexity of ECM and NFS, $C_{ECM}(k)$ and $C_{NFS}(k)$, as reported in section 3.2.

k	$\log C_{ECM}(k)$	$\log C_{NFS}(k)$	$123 + 3\log k$
512	62	87	150
1024	93	117	153
2048	139	156	156
2560	**158**	**178**	**156**
3072	175	189	159
4096	207	208	159

Thus, if one believes that ECM or NFS are best possible, the reduction suggests to set parameter k to something above 2048. Accordingly, n is a at least a 4096-bit modulus. This shows that the reduction only provides a qualitative assurance that the scheme is secure and that it cannot be interpreted with the suggested parameters.

5 Conclusion

Based on our analysis, we believe that the cryptosystem HIME-1 is presumably secure, with the proposed parameters. However, based on the submission, we would not recommend the scheme as it is for the following reasons:

- The specification contains ambiguities and mistakes.
- The range of suggested parameters only guarantees security for a foreseeable period of time which is rather limited.
- The submission does not include an appropriate security analysis. Although we have been able to offer a proof of the security of the scheme against adaptive chosen-ciphertext attacks, it rests on very recent research and it appears a bit fresh to form the basis of a cryptosystem.
- The estimates following the proof just mentioned would not give any conclusive evidence, when interpreted with the proposed parameters.

References

[1] 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.

[2] 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.

[3] M. Bellare and P. Rogaway, Optimal asymmetric encryption - How to encrypt with RSA, Eurocrypt'94, Lecture Notes in Computer Science 950, (1995), 92–111.

[4] 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.

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

[6] S. Cavallar, B. Dodson, A. K. Lenstra, P. Leyland, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, and P. Zimmermann, Factorization of RSA-140 using the Number Field Sieve, Asiacrypt '99, Lecture Notes in Computer Science 1716 (1999), 195–207.

[7] 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

[8] D. Coppersmith, Finding a Small Root of a Univariate Modular Equation; Eurocrypt'96, Lecture Notes in Computer Science 1070, (1996), 178–189.

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

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

[12] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA–OAEP is Still Alive. Cryptology ePrint Archive 2000/061, `http://eprint.iacr.org/`.

[13] Specification of HIME-1 Cryptosystem, Hitachi Ltd, `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`

[14] Self Evaluation Report, HIME-1 Cryptosystem, Hitachi Ltd, `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`

[15] Test Data, HIME-1 Cryptosystem,
http://www.sdl.hitachi.co.jp/crypto/hime/index.html

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

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

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

[19] 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.

[20] R. L. Rivest, A. Shamir, L. M. Adleman, Cryptographic Communications System and Method, US patent 4 405 829, September 20, 1983 (filed 14/12/1977).

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

[22] V. Shoup. OAEP Reconsidered. Cryptology ePrint Archive 2000/060, http://eprint.iacr.org/.

[23] T. Takagi, Fast RSA-Type Cryptosystem Modulo $p^k q$, Crypto'98, Lecture Notes in Computer Science 1462, (1998), 318-326.