Evaluation Report on the **HIME-2 Cryptosystem**

1 Introduction

This document is an evaluation of the **HIME-2 Cryptosystem**. Our work is based on the analysis of documents [11, 12, 13]. 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-2 is based on the hardness of factoring. It is a modified Rabin encryption scheme and uses integers n of the form $n = \prod_{i=1}^{d} p_i$, where the p_i are prime numbers - approximately of the same size - such that $p_i = 3 \bmod 4$, and where d is a small integer > 2, typically $d = 4$. The basic function on which HIME-2 is based is defined by

$$f(x) = x^2 \bmod n$$

We have the following:

Theorem 1 *Let y a non zero quadratic residue modulo n. There are exactly 2^d elements x such that $0 < x < n$ and $f(x) = y \bmod n$.*

Proof: This is well known. We have, for each i,

$$x^2 = y \bmod p_i$$

This equation has two square roots. Since p_i is congruent to 3 modulo 4, one is computed as $x_i = y^{\frac{p_i+1}{4}} \bmod p_i$ and the other by $-x_i \bmod p_i$. Using Chinese remaindering, we come out with a choice of 2^d possible values for $x \bmod n$.

Before explaining how the HIME-2 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.

Let k be the size in bits of integer n. The key generation algorithm $\mathcal{K}(1^k)$ of the HIME-2 Cryptosystem produces n as a product $n = \prod_{i=1}^{d} p_i$ of d odd primes p_i such that $p_i = 3 \bmod 4$ and d is any integer strictly greater than 2. The public key pk is the modulus n and thus defines the above function f. The secret key sk is the list of prime factors $(p_i)_{1 \le i \le d}$, which helps inverting f. The message space is $\mathcal{M} = \{0,1\}^{k-k_0-k_1}$, where k_0, k_1 are appropriate parameters. The encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$ uses two hash functions G and H:

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

It takes $m \in \mathcal{M}$ and a random value $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$, and computes a k-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 $c = f(x)$, with $x = a \| b \in \{0,1\}^k$. The decryption algorithm $\mathcal{D}_{\mathsf{sk}}(c)$ extracts the 2^d square roots of c, $x_i = a_i \| b_i$ and computes:

$$r_i = b_i \oplus H(a_i) \text{ and } M_i = a_i \oplus G(r_i)$$

It checks whether M_i is correctly formatted with its k_1 trailing bits zero and returns the $k - k_0 - k_1$ leading bits, if it finds such a properly formatted M_i. Otherwise it returns "Reject".

We refer to [11] for a precise definition of G and H and for an accurate choice of the parameters.

2.2 Comments on the specification

Document [11] needs further editing. There are ambiguities. For example, the restriction $p_i = 3 \bmod 4$ is not mentioned in the description of the algorithm (section **2.2.1.** *Key generation*) but only in the implementation section (**3.2.1.** *Key generation*). Theorem 1 above is true, regardless of this condition. However, the computation of the square root of y modulo p_i as $x_i = y^{\frac{p_i+1}{4}} \bmod p_i$ depends on the restriction: it is still possible to compute square roots in other cases but the algorithm is notably more intricate and, presumably, not appropriate in the present context. Also, the key generation algorithm (section **3.2.1** of document [11]) is vague: it is explained that four prime numbers p_i, with 256 bits are generated but nothing guarantees that $n = \prod_{i=1}^{d} p_i$ is actually 1024 bit long (it could be slightly less). There are inconsistencies. For example, notation φ is introduced at the end of section **2.2.3**, as a ring isomorphism defined on the product

$$\mathbb{Z}_p \oplus \mathbb{Z}_q$$

whereas p, q are undefined in the document and φ applies to d-tuples. In the same vein, notation

$$\varphi(x, y, m, n, z)$$

is used in a totally inconsistent manner throughout section **3** (*Implementation*). It is clear from the use of φ in the pseudo-code of section **3.2.3.** *Decryption*, that m should be the modulus related to x, n the modulus related to y and z should be $n^{-1} \bmod m$. This is not what appears in section **3.1.** *Symbols*, where there is a mismatch between m and n. A different mismatch also appears in section **3.3.6.** *Chinese remainder theorem*, even though the roles of m and n are swapped. Most ambiguities and errors can be corrected in a straightforward manner by cryptography experts. However, they might induce errors from implementors.

There are more serious concerns. Exception handlings in the decryption operation are simply not addressed. The pseudo-code of section **3.2.3** correctly covers the 16 possible square roots that can be found when $d = 4$. However, the formula

$$x = (m0^{k_1} \oplus G(r))||(r \oplus H(m0^{k_1} \oplus G(r)))$$

that defines x from m by OAEP formatting, outputs an element of bit-length k. Such element can be $> n$. Still, no mention of this case is made. Actually, if values of x which are $> n$ but $< 2n$ are allowed, there are 2^d additional values to test, obtained from the original 2^d values by adding n. It is possible to avoid considering such values by taking $k = |n| - 1$ but this is not proposed in the specification. Even in this case, incorrect ciphertexts are not properly handled by the decryption pseudo-code since the last trial value of m will be output anyway. There should be a rejection case included somewhere, which deals with the situation where all values of w returned by calls to the $convert^{-1}$ function are non zero. Such mistakes are particularly embarrassing as they

might pollute the subtle security arguments that have to be provided in favor of the cryptosystem. As a result, we have found impossible to carry our security analysis with the proposed specification of the scheme, as it is in document [12]. We have chosen to set $k = |n| - 1$ instead of $k = |n|$, so that

$$2^k \leq n < 2^{k+1}$$

Accordingly, we have restricted the range of f.

$$\begin{array}{rcl} f : \{0,1\}^k & \longrightarrow & \mathbb{Z}_n^\star \\ x & \longmapsto & x^2 \bmod n \end{array}$$

We will keep the convention $k = |n| - 1$, throughout our report.

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. As explained in section 2.2, we use $k = |n| - 1$ instead of $k = |n|$, as proposed by the submission.

3.1 Complexity-theoretic arguments

On page 5 of document [11], it is claimed that

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

and, in theorem 1.1 of document [12], its is stated that,

> HIME-2 is semantically secure against the adaptive chosen ciphertext attack, 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 one-to-one 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 not one-to-one, in view of theorem 1 above. Also, the range of the basic function is the set of quadratic residues modulo n and does not equal its domain. Thus, the wording *permutation* is improper.

Even though the word permutation is misleading, the basic function f is hard to invert under the assumption of the difficulty of factoring. More formally:

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

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

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

where probabilities include the internal random tape of the probabilistic machine $\mathcal{A}$. We use the above adversary $\mathcal{A}$ to factor n, by describing a Turing Machine $\mathcal{B}$ that makes calls to $\mathcal{A}$:

1. $\mathcal{B}$ chooses $x \stackrel{R}{\leftarrow} \{0,1\}^k$
2. $\mathcal{B}$ checks whether $\gcd(x, n) \notin \{1, n\}$, in such case, it returns the non-trivial gcd
3. $\mathcal{B}$ computes $y \leftarrow f(x)$;
4. $\mathcal{B}$ runs $\mathcal{A}$ on y, and gets z;
5. $\mathcal{B}$ returns $\gcd(n, z - x)$.

Let us study the probability that $\mathcal{B}$ returns a factor of n:

$$\begin{aligned} \nu &= \Pr[p \leftarrow \mathcal{B}] \\ &= \Pr[x \stackrel{R}{\leftarrow} \{0,1\}^k, y \leftarrow f(x), z \leftarrow \mathcal{A}(y) : x \in \mathbb{Z}_n^\star \wedge f(z) = y \wedge z \neq \pm x] \\ &\quad + \Pr[x \stackrel{R}{\leftarrow} \{0,1\}^k, y \leftarrow f(x), z \leftarrow \mathcal{A}(y) : x \notin \mathbb{Z}_n^\star \cup \{0\}]. \end{aligned}$$

For any quadratic residue $y \in \mathbb{Z}_n^\star$, we let $SQRT(y)$ denote the set of square roots of y in $\mathbb{Z}_n^\star$. For any x in $SQRT(y)$, $-x = n - x$ is also in $SQRT(y)$ and one of them is in $\{0,1\}^k$. Thus, for any quadratic residue y,

$$2^d \geq s = \#SQRT(y) \cap \{0,1\}^k \geq 2^{d-1}.$$

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

$$\begin{aligned} \Pr[f(z) = y \wedge z \neq \pm x \,|\, x \in \mathbb{Z}_n^\star] &\geq \Pr[f(z) = y \,|\, x \in \mathbb{Z}_n^\star] \times \begin{cases} \frac{s-2}{s} & \text{if } \{x, -x\} \subset \{0,1\}^k \\ \frac{s-1}{s} & \text{if } -x \notin \{0,1\}^k \end{cases} \\ &\geq \Pr[f(z) = y \,|\, x \in \mathbb{Z}_n^\star] \times \left(1 - \frac{2}{s}\right) \\ &\geq \Pr[f(z) = y \,|\, x \in \mathbb{Z}_n^\star] \times \left(1 - \frac{2}{2^{d-1}}\right) \end{aligned}$$

We get:

$$\begin{aligned}
\varepsilon &< \mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[z \in \{0,1\}^k \wedge f(z) = y] \\
&\leq \Pr[f(z) = y \wedge x \in \mathbb{Z}_n^\star] + \Pr[f(z) = y \wedge x \notin \mathbb{Z}_n^\star \cup \{0\}] + \Pr[f(z) = y \wedge x = 0] \\
&\leq \Pr[f(z) = y \wedge x \in \mathbb{Z}_n^\star] + \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] + \frac{1}{n}
\end{aligned}$$

from which we can bound ν:

$$\begin{aligned}
\nu &= \Pr[x \in \mathbb{Z}_n^\star \wedge f(z) = y \wedge z \neq \pm x] + \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] \\
&\geq \Pr[x \in \mathbb{Z}_n^\star \wedge f(z) = y] \times \left(1 - \frac{2}{2^{d-1}}\right) + \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] \\
&\geq \left(\varepsilon - \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] - \frac{1}{n}\right) \times \left(1 - \frac{2}{2^{d-1}}\right) + \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] \\
&\geq \left(\varepsilon - \frac{1}{n}\right) \times \left(1 - \frac{2}{2^{d-1}}\right) + \Pr[x \notin \mathbb{Z}_n^\star \cup \{0\}] \times \frac{2}{2^{d-1}} \geq \left(\varepsilon - \frac{1}{n}\right) \times \left(1 - \frac{1}{2^{d-2}}\right)
\end{aligned}$$

Therefore, the probability of success of $\mathcal{B}$ is greater than $(\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) - 1/n) \times (1 - 2^{-d+2})$. This finishes the proof.

What is crucial in the above is precisely the fact that $f(x)$ has more than one preimage and actually more than two. Finding, with significant probability, a preimage of $f(x)$ different from $\pm x$ allows to find a factor of n. We note that applying the same reduction repeatedly allows to factor n, but we do not include an exact analysis of the resulting 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 even if there is a tiny 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 $n = \prod_{i=1}^{d} p_i$. 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 two families, according to their sensitivity to the size of the factors.

Before entering into a more precise discussion, let us mention that the idea is of having more than two factors in an RSA (or Rabin) modulus is quite old. Thus, it duly appears in the RSA patent (see [16]). 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 $p_i - 1$ has a large prime factor r_i (document [11], page 9).

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 [14]). The limit used was $B = 15,000,000$.

In a recent paper [4], 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 [17]). 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 [9]). 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 ([6]). 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 [15]). In [6], 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 [4] and produces $Y = 2018$ for $D = 309$, i.e. for a 1024 bit modulus, as proposed in HIME-2.

3.2.3 Conclusion

Based on current estimates, it appears that the proposed parameters for HIME-2 should remain secure for at least thirteen years. Surprisingly, the use of moduli with four 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. Using 1024 bit moduli with six prime factors, as mentioned on page 7 of document [11], should certainly be avoided as it is already insecure today in view of the current ECM record.

4 Security Analysis

The self-evaluation report [12] does not includes a serious security analysis. It simply refers to [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 [18, 10]). We have to check whether or not the security analysis can be performed for HIME-2.

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

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

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* [8]. 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-2 scheme is IND-CCA in the random oracle model, based on the assumption that factoring is hard. More precisely, we 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 $\mathcal{B}$ inverting f, with success probability ε' and within time bound t' where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{4} \times \left(1 - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}\right) - \frac{q_G}{2^{k-3}} - q_D \cdot 2^d \times \left(\frac{1}{2^{k_1-1}} + \frac{q_G+1}{2^{k_0}}\right) \\ t' &\leq t + (q_D+1) \cdot q_H \cdot (\tau + \gamma) \end{aligned}$$

where τ is the time needed to execute the encryption algorithm, which is bounded by $\mathcal{O}(k^2)$ and γ is the time needed to execute the Coppersmith root finding algorithm from [7].

To prove the above, we follow [3, 10]: 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 f, defined from the public key
2. then $\mathcal{B}$ is given $y \leftarrow f(x)$, for $x \stackrel{R}{\leftarrow} \{0,1\}^k$, for which it wants to find a preimage; let x_i, $i = 1, \ldots, s$, denote the square roots of y in $\{0,1\}^k$. Thus, s is the number of possible preimages of y, $s \leq 2^d$. Each x_i can be split into $x_i = \alpha_i \| \beta_i$.
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 st. It chooses a random bit b, and then defines $C \leftarrow y$, to be a ciphertext of m_b
4. $\mathcal{B}$ runs $A_2(C, st)$ 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. Here, the simulation departs form [3]. It uses a randomly chosen index $\kappa \in \{1, \ldots, 2^d\}$. This index lies in $\{1, \ldots, s\}$ with probability greater than $s/2^d$, which is greater than $1/2$. The index is used to select a preimage of y, to be used by the simulator, by means of a counter c originally set to $c = 0$. Oracle calls are treated 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 $y = f(z)$. If for some δ this relation holds, then one preimage of y has been found. For each such preimage, one increases c by one, $c = c + 1$. If $c = \kappa$, 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 y and κ are random. 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 $y = f(z)$. If for some γ this relation holds, then f 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. In order make the analysis easier, we add extra steps to the simulation, querying H and G respectively on α_i and $H(\alpha_i) \oplus \beta_i$. Provided κ is $\leq s$, this allows to define the random variable x_κ, as the κ-th element of the list of preimages which appears in the extended computation as $\delta||(\gamma \oplus H(\delta))$, where γ is a query to G and δ a query to H. To keep notations simple, we write x in place of x_κ and α, β in place of α_κ, β_κ. We wish to be consistent with the assumption that the simulator uses x for encrypting m_b, which defines the corresponding random tape r as a random variable

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

It can be seen that this simulation is perfect, except in rare cases, and that we eventually find the preimage of y under f, from the list of queries asked to G and H. However, the complexity of this simulator is the complexity exceeds the running time of the adversary by an additional term corresponding to performing $q_G \times q_H$ squarings, which is huge. Following Shoup [18], we can improve the reduction: we start with an empty list $\mathcal{S}$, and a counter c initially set to 0.

- For a fresh query δ to H, output a random value H_δ. Additionally, run the Coppersmith root finding algorithm [7] to find, if it exists, an integer $u < 2^{k_0}$ such that $y = f(\delta \| u)$. If such a u exists, then add the triple $(\delta, u, u \oplus H_\delta)$ to the set $\mathcal{S}$.

- For a fresh query γ to G, if for some α', β', the triple $(\alpha', \beta', \gamma) \in \mathcal{S}$, increase c by one, $c = c + 1$. If $c = \kappa$, one can still correctly simulate G, by answering $G_\gamma = \alpha' \oplus m_b 0^{k_1}$. This is indeed a uniformly distributed value since y and κ are random, and thus α' is random. Otherwise, output a random value G_γ.

This simulation is similar to the previous one, but its complexity is less. Its computational load just exceeds the running time of the adversary by q_H executions of the Coppersmith root finding algorithm, and as many squarings to check to validity of the result.

Let AskG denote the event that query r has been asked to G (by A_1 or A_2, not taking the additional steps into account), and, similarly, let AskH be the event that query α has been asked to H. Also, let FAskH be 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 $\alpha \oplus m_1 0^{k_1}$

- 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 $G_r \leftarrow \delta \oplus m_b 0^{k_1} = \alpha \oplus m_b 0^{k_1}$

- and by κBad the event that κ is strictly greater than s.

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

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

Note that event κBad is independent of the other two, and that

$$\frac{1}{2} \leq \mathsf{pr}[\neg\kappa\mathsf{Bad}] = \frac{s}{2^d} \leq 1 \qquad 0 \leq \mathsf{pr}[\kappa\mathsf{Bad}] = 1 - \frac{s}{2^d} \leq \frac{1}{2}$$

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, with κ implicitly defined from the assumption that x_κ is used to encrypt m_b, since $\neg$AskG implies $\neg$(FBad$\vee$GBad), and it is clearly independent of b since b is undefined until the choice for $G(r)$ is set (in the additional steps, once b' has been output). 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:

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

Switching to probabilities in the simulated game, which is perfect unless FBad or GBad happens, or $\kappa > s$, we get:

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

Remark that

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

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 $\neg(\mathsf{FBad} \vee \mathsf{GBad})$), that the answer has

been either $\alpha \oplus m_0 0^{k_1}$ or $\alpha \oplus m_1 0^{k_1}$, plus the event that $\kappa \leq s$. This probability is less than

$$q_G \cdot 2^{-k_0} \times 2 \cdot 2^{-k+k_0} \times \mathsf{pr}[\neg\kappa\mathsf{Bad}] = q_G \cdot 2 \cdot 2^{-k} \times \mathsf{pr}[\neg\kappa\mathsf{Bad}] \leq q_G \cdot 2^{-k+1}$$

Therefore,

$$\mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \mid \neg\mathsf{Bad}] \geq \varepsilon/2 - q_G \cdot 2^{-k+1}/\mathsf{pr}[\neg\mathsf{Bad}]$$

and thus,

$$\begin{array}{rcl} \mathsf{pr}[\mathsf{AskG} \wedge \mathsf{AskH}] & \geq & \mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \wedge \neg\mathsf{Bad}] = \mathsf{pr}[(\mathsf{AskG} \wedge \mathsf{AskH}) \mid \neg\mathsf{Bad}] \times \mathsf{pr}[\neg\mathsf{Bad}] \\ & \geq & \varepsilon/2 \times \mathsf{pr}[\neg\mathsf{Bad}] - q_G \cdot 2^{-k+1} \end{array}$$

To conclude, we just have to evaluate the probability

$$\begin{array}{rcl} \mathsf{pr}[\neg\mathsf{Bad}] & = & 1 - \mathsf{pr}[\mathsf{Bad}] = 1 - \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \vee \kappa\mathsf{Bad}] \\ & \geq & 1 - (\mathsf{pr}[(\mathsf{FBad} \vee \mathsf{GBad}) \wedge \neg\kappa\mathsf{Bad}] + \mathsf{pr}[\kappa\mathsf{Bad}]) \\ & \geq & 1 - \mathsf{pr}[\kappa\mathsf{Bad}] - \mathsf{pr}[(\mathsf{FBad} \vee \mathsf{GBad}) \wedge \neg\kappa\mathsf{Bad}] \\ & \geq & \mathsf{pr}[\neg\kappa\mathsf{Bad}] - \mathsf{pr}[(\mathsf{FBad} \vee \mathsf{GBad}) \wedge \neg\kappa\mathsf{Bad}] \end{array}$$

We set $\pi = \mathsf{pr}[(\mathsf{FBad} \vee \mathsf{GBad}) \wedge \neg\kappa\mathsf{Bad}]$. We have:

$$\begin{array}{rcl} \pi & = & \mathsf{pr}[\neg\kappa\mathsf{Bad}] \times \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\kappa\mathsf{Bad} \wedge \neg\mathsf{FAskH}] \times \mathsf{pr}[\neg\mathsf{FAskH}] \\ & & +\mathsf{pr}[\neg\kappa\mathsf{Bad}] \times \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\kappa\mathsf{Bad} \wedge \mathsf{FAskH}] \times \mathsf{pr}[\mathsf{FAskH}] \\ & = & \mathsf{pr}[\neg\kappa\mathsf{Bad}] \times (\mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\kappa\mathsf{Bad} \wedge \neg\mathsf{FAskH}] + \mathsf{pr}[\mathsf{FAskH}]) \end{array}$$

and thus,

$$\mathsf{pr}[\neg\mathsf{Bad}] \geq \mathsf{pr}[\neg\kappa\mathsf{Bad}] \times (1 - \mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\kappa\mathsf{Bad} \wedge \neg\mathsf{FAskH}] - \mathsf{pr}[\mathsf{FAskH}])$$

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

Secondly, one observes that the conditional events FBad or GBad, knowing that $\neg\mathsf{FAskH}$ and $\neg\kappa\mathsf{Bad}$ hold, 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: $\mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\mathsf{FAskH}] \leq q_G \cdot 2^{-k_0}$.

$$\mathsf{pr}[\mathsf{FBad} \vee \mathsf{GBad} \mid \neg\kappa\mathsf{Bad} \wedge \neg\mathsf{FAskH}] \leq q_G \cdot 2^{-k_0}$$

Thus, since $1/2 \leq \mathsf{pr}[\neg\kappa\mathsf{Bad}]$,

$$\mathsf{pr}[\neg\mathsf{Bad}] \geq \frac{1}{2} \times \left(1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0}\right)$$

Finally,

$$\mathsf{pr}[\mathsf{AskH}] \geq \mathsf{pr}[\mathsf{AskG} \wedge \mathsf{AskH}] \geq \frac{\varepsilon}{4} \times \left(1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0}\right) - q_G \cdot 2^{-k+1}$$

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 (δ, H_δ) obtained from H. Then, for each δ, it runs the Coppersmith root finding algorithm [7] to find, if it exists, an integer $u < 2^{k_0}$ such that $c' = f(\delta \| u)$. For such a pair (δ, u) it defines

$$\gamma = u \oplus H_\delta \text{ and } M = G(\gamma) \oplus \delta,$$

where $G(\gamma)$ is computed using above G-simulation. Then it checks whether, furthermore, $[M]_{k_1} = 0^{k_1}$, 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 from the fact that f is at most 2^d-to-one: due to the randomness of G, the probability to have two possible plaintexts is less than $2^d/2^{k_1}$: with probability greater than $1 - 2^{d-k_1}$, at most one M is selected, and its trailing bits returned as the plaintext.

The ciphertext c' can be written $c' = f(a'_i \| b'_i)$, for $i = 1, \ldots, s$, and for each i, we let $r'_i = H(a'_i) \oplus b'_i$ (we use dashed variables for the ciphertext to decrypt). We denote by AskG' the event that some query r'_i has been asked to G, and by AskH' the event that some query a'_i has been asked to H. As already seen,the simulation can only fail by rejecting a valid ciphertext. Let us denote by Fail such an event:

$$\begin{aligned}
\mathsf{pr}[\mathsf{Fail}] &= \mathsf{pr}[\mathsf{Fail} \wedge \neg\mathsf{AskH}' \wedge \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{Fail} \wedge \neg\mathsf{AskH}' \wedge \mathsf{AskG}'] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskH}'] \\
&\geq \mathsf{pr}[\mathsf{Fail} \,|\, \neg\mathsf{AskH}' \wedge \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{AskG}' \,|\, \neg\mathsf{AskH}'] + \mathsf{pr}[\mathsf{Fail} \wedge \mathsf{AskH}'] \\
&\geq \mathsf{pr}[\mathsf{Fail} \,|\, \neg\mathsf{AskH}' \wedge \neg\mathsf{AskG}'] + \mathsf{pr}[\mathsf{AskG}' \,|\, \neg\mathsf{AskH}'] + 2^{d-k_1}
\end{aligned}$$

We argue that, if r'_i has not been asked to G, and $r'_i \neq r$, the probability that $[a'_i \oplus G(r'_i)]_{k_1} = 0^{k_1}$ is less than 2^{-k_1}. If furthermore a'_i has not been asked to H, then the probability for r'_i to be equal to r is less than 2^{k_0}. On the other hand, the probability to have asked $G(r'_i)$, without having asked $H(a'_i)$, which leaves r'_i random, is less than $q_G \cdot 2^{-k_0}$. Granted this, one can conclude that

$$\Pr[\mathsf{Fail}] \leq 2^d \times \left(2^{-k_1} + (q_G + 1) \cdot 2^{-k_0}\right) + 2^{d-k_1}$$

and, as a consequence, all decryption queries are correctly simulated with probability greater than

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

Thus, having given the simulator $\mathcal{B}$ access to the plaintext-extractor, we see that $\mathcal{B}$ perfectly simulates the interactions with a chosen-ciphertext adversary, with negligible

exceptions. More accurately, the probability for such a simulator $\mathcal{B}$ to output x is greater than

$$\frac{\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A})}{4} \times \left(1 - q_G \cdot 2^{-k_0} - q_H \cdot 2^{-k+k_0}\right) - \frac{q_G}{2^{k-1}} - q_D \cdot 2^d \times \left(\frac{1}{2^{k_1-1}} + \frac{q_G+1}{2^{k_0}}\right)$$

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 (δ, H_δ) obtained from H, to apply Coppersmith algorithm and to compute, for each input/output pair to this algorithm, say (δ, λ)

$$a = \delta, \gamma = \lambda \oplus H_\delta, M = G(\gamma) \oplus \delta$$

as well as $f(\delta \| \lambda)$. Proper bookkeeping allows to do this only once for each pair, and each target ciphertext (the challenge ciphertext, and the various queried ciphertext). Each operation is linear in k, except for computing $f(a\|b)$ and running the Coppersmith algorithm, whose complexities are denoted by τ and γ respectively.

Ignoring the linear terms, the time complexity of the overall reduction is

$$t + (q_D + 1) \cdot q_H \cdot (\tau + \gamma)$$

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 $\mathcal{A}$ 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 algorithm $\mathcal{B}$ able to output a proper factor of any integer $n = p_1 \dots p_d$, produced by the key generation algorithm of HIME-2, with success probability ε' and within time bound, t' where*

$$\begin{aligned} \varepsilon' &\geq \frac{\varepsilon}{4} \times \left(1 - \frac{1}{2^{d-2}} - \frac{q_G}{2^{k_0}} - \frac{q_H}{2^{k-k_0}}\right) - \left(\frac{q_G}{2^{k-3}} + q_D \cdot 2^d \times \left(\frac{1}{2^{k_1-1}} + \frac{q_G+1}{2^{k_0}}\right) + \frac{1}{n}\right) \\ t' &\leq t + ((q_D + 1) \cdot q_H + 1) \cdot (\tau + \gamma) \end{aligned}$$

Indeed, combining both reductions, one obtain the following success probability:

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

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

4.6 Practical security estimates

We try to understand whether the reduction shown above is 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

$$
\begin{aligned}
\varepsilon' &\geq \frac{\varepsilon}{4} \times \left(1 - \frac{1}{4} - \frac{1}{2^{68}} - \frac{1}{2^{324}}\right) - \left(\frac{1}{2^{449}} + 2^{34} \times \left(\frac{1}{2^{127}} + \frac{1}{2^{68}}\right) + \frac{1}{2^{512}}\right) \approx \frac{3\varepsilon}{16} - \frac{1}{2^{34}} \\
t' &\leq t + 2^{90} \cdot k^2
\end{aligned}
$$

Therefore, an adversary able to learn one bit with advantage greater than $1/2$, within time less than $2^{90}k^2$, can be used to factor n within less than $2^{94}k^2$ (note that we use a rather optimistic estimate of the running time of Coppersmith root finding algorithm as $\mathcal{O}(k^2)$)).

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	$\log C_{ECM}(k)$	$\log C_{NFS}(k)$	$94 + 2\log k$
1024	62	87	114
2048	93	117	116
3072	**118**	**139**	**118**
4096	139	156	118

Thus, if one believes that ECM or NFS are best possible, the reduction suggests to set parameter k to something above 3072. 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-2 is currently 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] R. P. Brent, Some Parallel Algorithms for Integer Factorisation, Euro-Par 99, Lecture Notes in Computer Science 1685, (1999), 1–22.

[5] 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.

[6] 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

[7] D. Coppersmith, Finding a small root of a univariate unimodular equation, Eurocrypt'96, Lecture Notes in Computer Science 1070, (1996), 155–165.

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

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

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

[11] Specification of HIME-2 Cryptosystem, Hitachi Ltd, `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`

[12] Self Evaluation Report, HIME-2 Cryptosystem, Hitachi Ltd, `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`

[13] Test Data, HIME-2 Cryptosystem, `http://www.sdl.hitachi.co.jp/crypto/hime/index.html`

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

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

[16] 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).

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

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