Self-Evaluation Report

HIME(R) CryptoSystem

Updated, October, 2003

Hitachi, Ltd.

Abstract

This document specifies the public-key cryptosystem HIME(R). HIME(R) is based on a modular squaring (Rabin's public-key encryption scheme [34]) over $\mathbb{Z}_N$, where $N = p^d q$ (p and q are prime integers, and $d > 1$), and utilize the fast calculation method for decryption. With HIME(R), security is additionally enhanced by the OAEP converting method [3].

HIME(R) has the following exceptional features:

- It is proven to be semantically secure against an adaptive chosen-ciphertext attack (IND-CCA2) in the random oracle model under the factoring assumption of N.
- It has a very fast encryption speed.
- The decryption speed (1536 bits) is about two-and-a-half times faster than that of RSA-OAEP (1024 bits) [3].
- The plaintext space is sufficiently large.
- The amount of computation for the encryption and decryption increases only slightly compared with previous schemes, even if the size of N increases in the future.

HIME(R) is the very practical public-key encryption scheme that is provably secure under the factoring assumption. This document details the security of HIME(R) and its performance.

Contents

1 Background

Many public-key cryptosystems have been presented, and among them the RSA scheme is the most famous and is well used. Unfortunately, however, RSA scheme is not secure against an adaptive chosen-ciphertext attack and a concrete attack against an actual system was shown [4]. Thus, RSA must be utilized in secure environment that the active attack is not effective.

Many studies on provably security of public-key cryptosystems have been actively carried out since the early 1990's and many practical provably secure schemes have been presented.

Dolve, Dwork and Naor presented a cryptosystem that is IND-CCA2 using reasonable intractability assumption. However, their scheme is completely impractical inasmuch as it relies on general and expensive construction for a non-interactive zero-knowledge proof [13].

Bellare and Rogaway presented a method for converting public-key encryption schemes based on trapdoor permutation to be IND-CCA1 [3], called OAEP (Although at first it was believed that OAEP could convert such schemes to IND-CCA2 schemes, it has recently been pointed out that the converted schemes are not IND-CCA2 but IND-CCA1 [36]). Their method is very practical and its security can be demonstrated using two assumptions, i.e., the computational intractability of inverting the trapdoor permutation and the existence of ideal hash functions. That is, the proof of security is given in the *random oracle model*, and this is a heuristic proof.

Cramer and Shoup presented a practical public-key cryptosystem which is IND-CCA2 in the standard model [11]. The security of their scheme is based on the intractability of the Decisional Diffie-Hellman (DDH) problem.

Boneh presented the public-key encryption schemes Rabin-SAEP, Rabin-SAEP+ and RSA-SAEP+ which are obtained by applying SAEP or SAEP+ (simplified versions of OAEP or OAEP+[36]) to Rabin's scheme or RSA[6].

Next, we will classify the security of public-key cryptosystems.

Attacks on public-key cryptosystems are classified as follows:

- **Passive Attack**
  - **Chosen-Plaintext Attack (CPA)：** An adversary can always gain the ciphertext for her chosen plaintext by sending the plaintext to an encryption oracle. Then the adversary attacks the given target ciphertext (An adversary can always wage this attack on public-key cryptosystems because the encipher keys are published.).
- **Active Attack**
  - **Non-Adaptive Chosen-Ciphertext Attack (CCA1)：**An adversary can gain the plaintext for her chosen ciphertext by sending this ciphertext to a decryption oracle before the target ciphertext is given. Then the adversary attacks the given target ciphertext.
  - **Adaptive Chosen-Ciphertext Attack (CCA2)：** An adversary can always gain the plaintext for all but her target ciphertext by sending ciphertext to a decryption oracle. Then the adversary attacks the given target ciphertext.

The above description shows that CCA1 is a stronger attack than CPA, and CCA2 is a stronger attack than CCA1.

Security levels of public-key cryptosystems are classified as follows.

- **One-Wayness (OW) :** It is hard for adversaries to invert the encryption function.
- **Semantic Security / Indistinguishability (IND) :** It is hard for adversaries to compute partial information about the plaintext from its ciphertext.
- **Non-Malleability (NM) :** It is hard for adversaries to compute a relation for R and the ciphertexts $y_i = E(x_i)$ $(1 \leq i \leq k)$ which satisfy $R(x, x_1, x_2, \ldots, x_k)$ for the ciphertext $y = E(x)$, where E is an encryption function.

Now, we can form {security level}-{attack} pairs. For example, if we say that a public-key cryptosystem is NM-CCA2, it means that the cryptosystem is non-malleable against an adaptive chosen-ciphertext attack. Figure 1 shows the relation among these pairs *. Here, $A \rightarrow B$ denotes that if a public-key cryptosystem is A, then it is certainly B. The $A \not\rightarrow B$ denotes its denial. The important point is that IND-CCA2 and NM-CCA2 are equivalent. Therefore, public-key cryptosystems that are IND-CCA2 or NM-CCA2 will have the highest level of security.

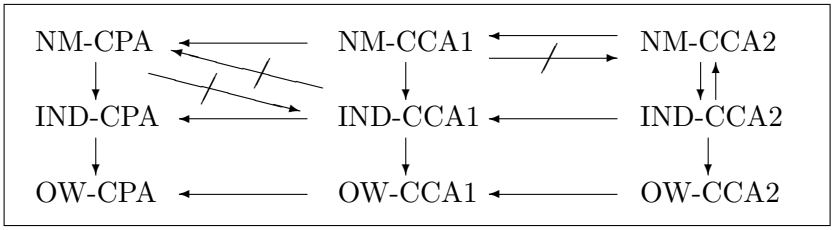

Figure 1: Relation among definitions of security for public-key cryptosystems.

The main objective of this document is to evaluate the public-key cryptosystem HIME(R). The design policy of HIME(R) and overview is described in Section 2. The basic algorithm of HIME(R) is given in Section 3, its security in Section 4 and its performance in Section 5.

*This relation is discussed in [1].

2 Design Policy and Overview

The design policy of HIME(R) is as follows:

(1) Security : It can be proven to be secure in the sense of IND-CCA2 under the assumption of the intractability of primitive problems (whose computational intractability is expected under the enough studies, such as the factoring problem or the discrete logarithm problem).

(2) Efficiency :

(2-1) Both encryption and decryption speeds are fast.

(2-2) The ratio of a plaintext and a ciphertext "(Plaintext)/(Ciphertext)" is not small.

(2-3) The plaintext space is sufficiently large.

(2-4) It can be mounted with a small memory size (including public key and private key sizes).

In terms of security, we believe that the factoring problem or the discrete logarithm problem are almost ideal as a number theoretic assumption of cryptosystems, because with sufficient study their computational intractability can be taken for granted [17, 25, 26]. Furthermore, there are two categories in number theoretic assumptions that are well utilized in the practical cryptosystems, i.e.:

Factoring-based: Factoring problem, RSA problem, Quadratic residue problem, etc,

Discrete-Logarithm-based: Discrete logarithm problem, Computational Diffie-Hellman problem, Decisional Diffie-Hellman problem, etc,

and the factoring problem and the discrete logarithm problem are the most intractable problems in each category.

In constructing HIME(R), we focused on the modular square function (Rabin's encryption function), because it is well known that inverting the encryption function on $\mathbb{Z}_N$ is as intractable as the factoring of N, where $N = pq$ (p and q are prime numbers). Another reason is that it has fast encryption speed. However, the following problems were encountered:

(P-1) The modular square function is not one-way trapdoor permutation, i.e., the decryption is not done uniquely.

(P-2) Rabin's scheme is not secure against a chosen-ciphertext attack.

(P-3) The decryption speed is not fast (i.e., it is as same as that of RSA).

In HIME(R), we utilize OAEP [3] to solve the problems (P-1) and (P-2). Owing to OAEP, we can get the probabilistically uniqueness of the decryption (cf. Section 3.4) and prove that it is secure in the sense of IND-CCA2 in the random oracle model by using

Coppersmith's algorithm (cf. Sections 4.4 and 4.5). Note that since OAEP was designed for the public-key encryption schemes that are based on one-way trapdoor permutations [†] and HIME(R) is based on the modular square function $f(x) = x^2 \bmod N$ (N is a composite number) that is not the trapdoor permutation, it was necessary to show if OAEP could apply to HIME(R) and it was possible to prove its security. We used this idea, applying OAEP to Rabin's scheme to solve (P-1) and (P-2), in HIME-2 [21]. After that the same idea was used in Rabin-SAEP and Rabin-SAEP+ even though the padding method differs from OAEP. We think that OAEP has the following advantage compared with SAEP and SAEP+.

(1) The security depending on ideal hash functions should be made to mitigate in consideration of application to an actual system: In actual systems, the ideal hash functions are replaced by practical hash functions, such as SHA [30], because no ideal hash functions exist. Therefore, the proof of security in random oracle model cannot transfer to the real world, and it is important to analyze the security of provably secure schemes in the random oracle model in the real world. We believe that the security of SAEP and SAEP+ depends on the ideal hash functions more heavily than the security of OAEP and OAEP+ does. For example, suppose that the adversary can compute the first m_1 bits and the last m_2 of $f^{-1}(y)$, where $m_1 + m_2 < |N|/2$, y is a target ciphertext, and f is an encryption function of Rabin-SAEP, Rabin-SAEP+ or RSA-SAEP+. Then, it is impossible to apply Coppersmith's algorithm to compute the rest bits of $f^{-1}(y)$. Furthermore, suppose that the first m_1 bits of the output of the hash function H has bias in response to the last m_2 bits of the input. That is, the first m_1 bits of $H(x)$ can be computed with a probability of more than $1/2^{m_1}$ when the last m_2 bits of x are known. The adversary will then be able to guess a correct b with a probability of more than $1/2$ (cf. Definition 4.1). However, we think that the possibility of this occurring in OAEP and OAEP+ is less than that in SAEP and SAEP+ because the plaintext is doubly protected by the two hash functions G and H. From the above reason, we think that OAEP and OAEP+ are more secure padding methods than SAEP and SAEP+ in the real world. Therefore, we believe that our scheme has a higher security than Rabin-SAEP and Rabin-SAEP+ in the real world.

(2) The plaintext space should be taken largely: The main purpose of public-key cryptosystems is to distribute the data enciphering key of secret-key cryptosystems. However, there are many systems, such as SET, that the additional information, such as identity information, are attached with the data enciphering key. In HIME(R), we made it one of our design policy in considering such systems. The maximal lengths of SAEP and SAEP+ are respectively 256 bits and 384 bits, where the modulo N is 1024 bits. Hence, they are inferior than OAEP in this point (See Section 5.3 for details). We can thus nearly clear the above conditions (2-2) and (2-3) by using OAEP.

As OAEP+ has a demerit of requiring one more hash function than OAEP, we therefore believed that OAEP is the best method to apply HIME(R).

[†]It was believed that OAEP could convert a public-key encryption scheme that is induced from an one-way trapdoor permutation f to an IND-CCA2 scheme under the assumption of the intractability of computing f^{-1}. Recently, however, it was pointed out that OAEP could convert the scheme to IND-CCA1 scheme but exceptions for general f [36], where it was shown that RSA-OAEP is secure in the sense of IND-CCA2 [36, 16]. OAEP+ is proposed as the modified OAEP that solve the problem of OAEP [36]. In OAEP+, one more hash function is required than OAEP.

In HIME(R), we make $N = p^d q$ (p, q: prime numbers, $d > 1$) instead of $N = pq$ and utilize our calculation method over $\mathbb{Z}_N$ to solve (P-3). Previously, a modified RSA scheme was proposed that utilizes such N and applies the original calculation method to make the decryption speed of RSA faster. The original calculation method was done over $\mathbb{Z}_{p^d}$ after $\mathbb{Z}_N$ is divided into $\mathbb{Z}_{p^d}$ and $\mathbb{Z}_q$ by using the Chinese Remainder Theorem (CRT), and the calculated values on $\mathbb{Z}_{p^d}$ and $\mathbb{Z}_q$ were combined on $\mathbb{Z}_N$ by using CRT again. Our calculation method differs from this previous one in that ours require no calculation by using CRT. As a result, our method has the following advantages:

- It has less modular multiplications than the previous one (cf. Section 5.2).
- The actual decryption speed and mountaing size will be smaller than previous one because ours does not require Euclidean algorithm for CRT.

Although this difference is very small, it is expected that it will be non-negligible in smart card systems and in systems in which much decryption processing must be done at one time.

On the other hand, HIME(R) avoids the need for a hybrid scheme [‡] with a secret-key encryption scheme, meaning that solving (2-4) would require no secret-key encryption scheme to enable public-key encryption. Another problem with hybrid schemes is that they may require the use of two different secret-key cryptosystems in a single system, which would add to development costs. Key encapsulation mechanism (KEM) [37] is recently proposed, and it is for distribution of the data encryption key of secret-key encryption schemes. Note that ordinary public-key encryption schemes can also utilize as KEM, and that the schemes secure in the sense of IND-CCA2 satisfy the conditions that are required to accomplish the security of KEM (see the security notion of KEM in [37] for details).

From the above discussion, HIME(R) has almost ideal features as follows:

(H-1) It is proven to secure in the sense of IND-CCA2 in the random oracle model under the factoring assumption of N ($= p^d q$, $d > 1$).

(H-2) It has a very fast encryption speed.

(H-3) Its decryption speed (1536 bits) is about two-and-a-half times faster than that of RSA-OAEP (1024 bits).

(H-4) The plaintext space is sufficiently large.

(H-5) The amount of computation for the encryption and decryption increases only slightly compared with previous schemes, even if the size of N increases in the future.

The condition (H-5) is important for future considerations, although we did not adopt this condition in (2-1) $\sim$ (2-4). The processing ability of computers is increasing rapidly, then the key length must also increase to stay ahead. This increase in key length impairs the efficiency of encryption schemes. However, our scheme can be used well into the future, because it can achieve efficient encryption and decryption processing even if the key length increases (cf. Section 4.6 and 5).

We show the superiority of HIME(R) in Section 5.

[‡]EPOC-2 [8] is known as the factoring base hybrid scheme.

3 Basic Scheme

In this section, we present the basic scheme of HIME(R): After this, $|x|$ denotes a binary length of x.

3.1 Key Generation

(K-1) Choose large prime numbers p, q, such that $|p| = |q|$, $p \equiv 3 \pmod 4$, and $q \equiv 3 \pmod 4$.

(K-2) Choose an integer d with $d > 1$.

(K-3) Compute $N = p^d q$.

(K-4) Choose positive integers k_0, k_1 and n [§] such that $n = k - k_0 - k_1 - 1$ and $2k_0 < k$, where $|N| = k$.

(K-5) Choose the hash functions G and H such that

$$G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}, \qquad H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}.$$

Then we make

Private key: (p, q),

Public key: (N, k, k_0, k_1, G, H).

Note that $N/2 < 2^{k-1} < N < 2^k$. Although the above algorithm is given for general d, we strongly recommend that $d = 2$ be chosen at present (The efficiency of decryption can be increased by taking $d > 2$ when $|N| \geq 4096$). We give the details of the length of each parameter k_0, k_1 and k in Section 5.1.

3.2 Encryption

(E-1) For a message $x \in \{0,1\}^n$ with $\gcd(x, N) = 1$, choose the random number $r \in \{0,1\}^{k_0}$, and compute

$$X = (x0^{k_1} \oplus G(r))||(r \oplus H(x0^{k_1} \oplus G(r))).$$

(E-2) Compute

$$y = X^2 \bmod N.$$

Then, y is given as a ciphertext of x.

[§]Correctly, k_0, k_1 and n are positive integer valued functions of k.

3.3 Decryption

For the given ciphertext y,

(D-1) Check if y is a quadratic residue on $\mathbb{Z}_N$, namely check

$$y^{\frac{p-1}{2}} \equiv 1 \pmod{p} \qquad \text{and} \qquad y^{\frac{q-1}{2}} \equiv 1 \pmod{q}.$$

If y is not a quadratic residue, reject it.

(D-2) For $i, j \in \{0, 1\}$, compute

$$\gamma_0^{(i)} = (-1)^i y^{\frac{p+1}{4}} \bmod p, \qquad \gamma_1^{(j)} = \left((-1)^j y^{\frac{q+1}{4}} - x_0\right) p^{-1} \bmod q$$

and

$$\gamma_l^{(i,j)} = \frac{y - {\Gamma_{l-1}^{(i,j)}}^2 \bmod p^l q}{p^{l-1} q} \times (2\gamma_0^{(i)})^{-1} \bmod p \quad (2 \le l \le d),$$

where

$$\Gamma_1^{(i,j)} = \gamma_0^{(i)} + \gamma_1^{(j)} p \quad \text{and} \quad \Gamma_l^{(i,j)} = \Gamma_{l-1}^{(i,j)} + \gamma_l^{(i,j)} p^{l-1} q \quad (2 \le l \le d-1).$$

(D-3) For $i, j \in \{0, 1\}$, compute

$$X_{i,j} = \gamma_0^{(i)} + \gamma_1^{(j)} p + \sum_{l=2}^{d} \gamma_l^{(i,j)} p^{l-1} q.$$

Let those $X_{i,j}$ $(0 \le i, j \le 1)$ be X_1, X_2, X_3, X_4.

(D-4) Choose X_i such that $X_i \in \{0, 1\}^{k-1}$.

(D-5) For each X_i, compute $s_i \in \{0, 1\}^{n+k_1}$ and $t_i \in \{0, 1\}^{k_0}$ such that $X_i = s_i || t_i$.

(D-6) For each s_i and t_i, compute

$$r_i = H(s_i) \oplus t_i,$$

and compute

$$w_i = s_i \oplus G(r_i).$$

(D-7) For each w_i, compute $x_i \in \{0, 1\}^n$ and $z_i \in \{0, 1\}^{k_1}$ such that $w_i = x_i || z_i$, and output

$$\begin{cases} x_i & \text{if } z_i = 0^{k_1} \text{ for a unique } i \\ \text{“Reject”} & \text{otherwise,} \end{cases}$$

as the decryption of the ciphertext y.

3.4 Soundness of Decryption

Theorem 3.1. In the algorithm of HIME(R), the plaintext is correctly decoded from the valid ciphertext except with a negligible probability.

Proof. We first show that X_i $(1 \leq i \leq 4)$ are all square roots of y in $\mathbb{Z}_N$. If it is shown, there are at most four square roots of y in $\{0,1\}^{k-1}$.

We show this by induction on d. Note that any element x in $\mathbb{Z}_N$ $(N = p^d q)$ can be written by

$$x = \gamma_0 + \gamma_1 p + \sum_{i=2}^{d} \gamma_i p^{i-1} q \qquad (0 \leq \gamma_0, \gamma_2, \ldots, \gamma_{d-1} < p, \quad 0 \leq \gamma_1 < q),$$

and such γ_i is uniquely determined.

Let $d = 2$. Then, the element x in $\mathbb{Z}_{p^2q}$ can be written by $x = \gamma_0 + \gamma_1 p + \gamma_2 pq$ for some $\gamma_0, \gamma_1, \gamma_2 \in \mathbb{Z}$ $(0 \leq \gamma_0, \gamma_2 < p,\ 0 \leq \gamma_1 < q)$.

Suppose that $x^2 \equiv y \pmod{p^2 q}$. Then, we have

$$\begin{aligned} x^2 &\equiv (\gamma_0 + \gamma_1 p + \gamma_2 pq)^2 \\ &\equiv {\gamma_0}^2 + {\gamma_1}^2 p^2 + 2\gamma_0\gamma_1 p + 2\gamma_0\gamma_2 pq \equiv y \pmod{p^2 q}. \end{aligned} \tag{1}$$

And it follows that

$${\gamma_0}^2 \equiv y \pmod{p} \qquad \text{and} \qquad (\gamma_0 + \gamma_1 p)^2 \equiv y \pmod{q}.$$

Since p and q are Blum numbers, γ_0 and γ_1 can be computed as follows (after testing if $y \bmod p$ and $y \bmod q$ are quadratic residue on $\mathbb{Z}_p$ and $\mathbb{Z}_p$ respectively):

$$\begin{aligned} \gamma_0 &= y^{\frac{p+1}{4}} \bmod p \quad \text{or} \quad -y^{\frac{p+1}{4}} \bmod p, \\ \gamma_1 &= (y^{\frac{q+1}{4}} - \gamma_0)p^{-1} \bmod q \quad \text{or} \quad (-y^{\frac{q+1}{4}} - \gamma_0)p^{-1} \bmod q. \end{aligned}$$

Hence γ_0 is $\gamma_0^{(0)}$ or $\gamma_0^{(1)}$, and γ_1 is $\gamma_1^{(0)}$ or $\gamma_1^{(1)}$. Furthermore, γ_2 is induced from the equation (1) as follows:

$$\gamma_2 = \frac{y - (\gamma_0 + \gamma_1 p)^2 \bmod p^2 q}{pq} \times (2\gamma_0)^{-1} \bmod p.$$

Hence γ_2 is $\gamma_2^{(0,0)}$, $\gamma_2^{(1,0)}$, $\gamma_2^{(0,1)}$ or $\gamma_2^{(1,1)}$. Note that pq divides $y - (\gamma_0 + \gamma_1 p)^2 \bmod p^2 q$. We can also easily prove that y is a quadratic residue on $\mathbb{Z}_{p^2q}$ if and only if $y \bmod p$ and $y \bmod q$ are respectively quadratic residue on $\mathbb{Z}_p$ and $\mathbb{Z}_q$.

From the above, it was shown that x_0, x_1, x_2, x_4 are all square roots of y in $\mathbb{Z}_{p^2q}$.

Next, let $d > 2$. And assume that $\Gamma_{d-1}^{(i,j)}$ $(= \gamma_0^{(i)} + \gamma_1^{(j)} p + \sum_{l=2}^{d-1} \gamma_l^{(i,j)} p^{l-1} q)$ are all square roots of y in $\mathbb{Z}_{p^{d-1}q}$ for $0 \leq i, j \leq 1$. Suppose that

$$x^2 \equiv y \pmod{p^d q}, \tag{2}$$

for some $x \in \mathbb{Z}_{p^d q}$. Then, from the assumption, x can be written by

$$x = \Gamma_{d-1}^{(i,j)} + \gamma_d^{(i,j)}\, p^{d-1} q,$$

for some $\gamma_d^{(i,j)} \in \mathbb{Z}$ ($0 \leq \gamma_d^{(i,j)} < p$, $0 \leq i, j \leq 2$). And we have

$$x^2 \equiv (\Gamma_{d-1}^{(i,j)} + \gamma_d^{(i,j)} p^{d-1} q)^2 \equiv {\Gamma_{d-1}^{(i,j)}}^2 + 2\Gamma_{d-1}^{(i,j)} \gamma_d^{(i,j)} p^{d-1} q \equiv y \pmod{p^d q},$$

from the equation (2). Hence $\gamma_d^{(i,j)}$ can be obtained by

$$\gamma_d^{(i,j)} = \frac{y - {\Gamma_{d-1}^{(i,j)}}^2 \bmod p^d q}{p^{d-1} q} \times (2\gamma_0^{(i)})^{-1} \bmod p.$$

Note that $p^{d-1}q$ divides $y - {\Gamma_{d-1}^{(i,j)}}^2 \bmod p^d q$. We can also easily prove that y is a quadratic residue mod $p^d q$ if and only if $y \bmod p$ and $y \bmod q$ are respectively quadratic residue on $\mathbb{Z}_p$ and $\mathbb{Z}_q$, by induction.

Next, we consider the probability that the decryption fails. Let y be a (valid) ciphertext, and let x be the plaintext of y, namely, $y = X^2 \bmod N$ for $X = s||t$, where $s = x0^{k_1} \oplus G(r)$, $t = r \oplus H(s)$, $x \in \{0,1\}^n$, and $r \in \{0,1\}^{k_0}$. Let X' be a $k-1$ bits string such that $y \equiv {X'}^2 \pmod N$ and $X \neq X'$, and suppose that $X' = s'||t'$ for $s' = (x'_1||x'_2) \oplus G(r')$, $t' = r' \oplus H(s')$, $x'_1 \in \{0,1\}^n$, $x'_2 \in \{0,1\}^{k_1}$, and $r' \in \{0,1\}^{k_0}$. We define the following event to consider the probability that the decryption of y fails because of X':

FAIL-X' is true if $x'_2 = 0^{k_1}$.

Then we have

$$\begin{aligned}\Pr[\textbf{FAIL-}X'] &= \Pr[\textbf{FAIL-}X' \mid r = r'] \cdot \Pr[r = r'] + \Pr[\textbf{FAIL-}X' \mid r \neq r'] \cdot \Pr[r \neq r'] \\ &\leq \Pr[r = r'] + \Pr[\textbf{FAIL-}X' \mid r \neq r']. \end{aligned} \tag{3}$$

We first consider $\Pr[r = r']$. We have

$$\begin{aligned}\Pr[r = r'] &= \Pr[r = r' \mid s = s'] \cdot \Pr[s = s'] + \Pr[r = r' \mid s \neq s'] \cdot \Pr[s \neq s'] \\ &\leq \Pr[r = r' \mid s = s'] + \Pr[r = r' \mid s \neq s'].\end{aligned}$$

If $s = s'$ then $t \neq t'$ because $X \neq X'$. Since $r = t \oplus H(s)$ and $r' = t' \oplus H(s')$, we have $\Pr[r = r' \mid s = s'] = 0$. On the other hand, we have $\Pr[r = r' \mid s \neq s'] = 1/2^{k_0}$ because H is a random oracle. Therefore, we obtain

$$\Pr[r = r'] \leq \frac{1}{2^{k_0}}. \tag{4}$$

Next, we consider $\Pr[\textbf{FAIL-}X' \mid r \neq r']$. Since G is a random oracle, we have

$$\Pr[\textbf{FAIL-}X' \mid r \neq r'] = \frac{1}{2^{k_1}}. \tag{5}$$

We finally obtain

$$\Pr[\textbf{FAIL-}X'] \leq \frac{1}{2^{k_0}} + \frac{1}{2^{k_1}}$$

from (3), (4) and (5).

We have already shown that there are at most three candidates such as X' that cause a failure of the decryption. Hence the probability that the decryption fails is less than $1 - (1 - 1/2^{k_0} - 1/2^{k_1})^3$.

□

4 Security

We use notations and conventions of [1].

4.1 Probabilistic Algorithms

If A is a probabilistic algorithm, then $A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r. We let $y \leftarrow A(x_1, x_2, \ldots)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \ldots; r)$. If S is a finite set then $x \leftarrow S$ is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $s \leftarrow \alpha$ is a simple assignment statement.

4.2 Encryption Schemes

An asymmetric (i.e., public-key) encryption scheme is given by a triple of algorithms, $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$, where

- $\mathcal{K}$, the *key generation algorithm*, is a probabilistic algorithm that takes a security parameter $k \in \mathbb{N}$ and returns a pair (pk, sk) of matching public and private keys.
- $\mathcal{E}$, the *encryption algorithm*, is a probabilistic algorithm that takes a public key pk and a message $x \in \{0,1\}^*$ to produce a ciphertext y.
- $\mathcal{D}$, the *decryption algorithm*, is a deterministic algorithm that takes a private key sk and a ciphertext y to produce either a message $x \in \{0,1\}^*$ or a special symbol $\perp$ to indicate that the ciphertext was invalid.

We require that for all (pk, sk) which can be output by $\mathcal{K}(1^k)$, for all $x \in \{0,1\}^*$, and for all y that can be output by $\mathcal{E}_{pk}(x)$, we have that $\mathcal{D}_{sk}(y) = x$. We also require that $\mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ can be computed in polynomial time. As the notation indicates, the key are indicated as subscripts to the algorithms.

We say that Π is an encryption scheme in *the random oracle model* if the algorithms $\mathcal{E}$ and $\mathcal{D}$ can access to random oracles to produce a ciphertext and a message (or $\perp$) respectively.

4.3 Indistinguishability

We say that a function *epsilon* : $\mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c \geq 0$ there exists an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$.

Definition 4.1 (IND-ATK in the random oracle model).
Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $A = (A_1, A_2)$ be an adversary. For atk $\in$ {cpa, cca1, cca2} and $k \in \mathbb{N}$, let

$$\mathbf{Adv}^{\mathrm{ind-atk}}_{A,\Pi}(k) \stackrel{\mathrm{def}}{=} 2 \cdot \Pr\big[H \leftarrow \mathsf{Hash}\ ;\ (pk, sk) \leftarrow \mathcal{K}(1^k)\ ;\ (x_0, x_1, s) \leftarrow A_1^{H,\mathcal{O}_1}(pk)\ ; \\ b \leftarrow \{0,1\}\ ;\ y \leftarrow \mathcal{E}^H_{pk}(x_b)\ :\quad A_2^{H,\mathcal{O}_2}(x_0, x_1, s, y) = b\big] - 1,$$

where

If atk=cpa	then	$\mathcal{O}_1(\cdot) = \varepsilon$	and	$\mathcal{O}_2(\cdot) = \varepsilon$
If atk=cca1	then	$\mathcal{O}_1(\cdot) = D^H_{sk}(\cdot)$	and	$\mathcal{O}_2(\cdot) = \varepsilon$
If atk=cca2	then	$\mathcal{O}_1(\cdot) = D^H_{sk}(\cdot)$	and	$\mathcal{O}_2(\cdot) = D^H_{sk}(\cdot)$,

Hash denotes a set of all functions from some appropriate domain to appropriate range [¶], s denotes state information (possibly including pk) which the adversary wants to preserve, and D_{sk} denotes the decryption oracle. For the above, we insist that A_1 outputs x_0, x_1 with $|x_0| = |x_1|$. We say that Π *is secure in the sense of* IND-ATK *in the random oracle model* if A being polynomial-time implies that $\mathbf{Adv}^{\mathrm{ind-atk}}_{A,\Pi}(\cdot)$ is negligible.

4.4 Coppersmith's Algorithm

In this section, we present the fact by Coppersmith [10].

[**Coppersmith**] Let N be a large composite integer of unknown factorization. Let

$$f(x) = x^k + a_{k-1}x^{k-1} + \cdots + a_2x^2 + a_1x + a_0 \in \mathbb{Z}[x]$$

be a monic polynomial of degree k. Then, there is an efficient algorithm to find all $x_0 \in \mathbb{Z}$ such that

$$f(x_0) = 0 \pmod N \qquad \text{and} \qquad |x_0| < N^{1/k}.$$

We denote by $T_C(N, k)$ the running time of Coppersmith's algorithm when finding roots of a polynomial $f \in \mathbb{Z}[x]$ of degree k.

4.5 Proof of Security

A *composite number generator* $\mathcal{G}$ is a probabilistic polynomial time (PPT) algorithm such that $\mathcal{G}(1^k)$ outputs a composite number N, where $|N| = k$.

Definition 4.2. Let $\mathcal{G}$ be a composite number generator. We say that algorithm M succeeds in (t, ϵ)-*factoring* $\mathcal{G}(1^k)$ if

$$\Pr\left[N \leftarrow \mathcal{G}(1^k) \ : \ M(N) = (p_1, p_2, \ldots, p_d)\right] \geq \epsilon,$$

where $N = \prod_{i=1}^{d} p_i$ (each p_i is prime numbers), and, moreover, in the experiment above, M runs in at most t steps.

We simply say that *the factoring* $\mathcal{G}$ *is intractable* if there is no polynomial time algorithm M which succeeds in (t, ϵ)-factoring $\mathcal{G}(1^k)$ for the non-negligible ϵ.

Then we obtain the following theorem.

Theorem 4.1. HIME(R) is secure in the sense of IND-CCA2 in the random oracle model under the assumption of intractability of factoring $N\,(= p^d q)$.

[¶] These sets might change from scheme to scheme.

Let $\mathcal{G}$ be a composite number generator such that $N\,(=p^dq) \leftarrow \mathcal{G}(1^k)$. We assume that the distribution of N is the same as that of N with HIME(R).

The proof of Theorem 4.1 is immediately induced from the following theorem.

Theorem 4.2. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be the our encryption scheme with parameters k_0 and k_1, and let n be the associated plaintext length. Then, there exists an oracle machine U such that for each integer k the following is true. Suppose $A = (A_1, A_2)$ succeeds in $(t, q_D, q_G, q_H, \epsilon)$-breaking $\Pi(1^k)$ in the sense of IND-CCA2, namely,

$$2 \cdot \Pr[G, H \leftarrow \mathsf{Hash}\,;\ (pk, sk) \leftarrow \mathcal{K}(1^k)\,;\ (x_0, x_1, s) \leftarrow A_1^{G,H,\mathcal{D}_{sk}^{G,H}}(pk)\,;$$
$$b \leftarrow \{0,1\}\,;\ y \leftarrow \mathcal{E}_{pk}^{G,H}(x_b)\ :\ A_2^{G,H,\mathcal{D}_{sk}^{G,H}}(x_0, x_1, s, y) = b] - 1 \geq \epsilon,$$

where A runs for at most t steps, makes at most q_D queries to the decryption oracle, makes at most q_G queries to G, and makes at most q_H queries to H.

Then, $M = U^A$ succeeds in (t', ϵ')-factoring $\mathcal{G}(1^k)$, where

$$t' \leq t + q_H\, T_C(N, 2) + q_G\, q_H\, T_S(k) + \tilde{T}(k) + \mathcal{O}(k)$$
$$\epsilon' \geq \frac{1}{3}\left(\epsilon - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{2q_G + q_D}{2^{k_0}} - \frac{q_D}{2^{k_1}}\right)\left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D}.$$

Here, $T_S(k)$ denotes the running time of the encryption function $\mathcal{E}_{pk}(\cdot)$, and $\tilde{T}(k)$ denotes the running time of factoring N when an integer that has a commom factor with N is given. Recall that $T_C(N, k)$ denotes the running time of Coppersmith's algorithm when finding roots of a polynomial $f \in \mathbb{Z}[x]$ of degree k.

Proof. We first define the behavior of factoring algorithm M. M is given a composite number N $(= p^dq)$. It is trying to find the prime factor of N. The factoring algorithm M is defined as follows:

(0) An input to M is N, where $N \leftarrow \mathcal{G}(1^k)$.

(1) M chooses $s \in \{0,1\}^{n+k_1}$, $t \in \{0,1\}^{k_0}$ and $b \in \{0,1\}$ at random, and set $w = s||t$. M computes $y = w^2 \bmod N$.

(2) M initializes two lists, called its G-list and its H-list, to empty.

Then, M simulates the two stages of $A = (A_1, A_2)$ as indicated in the next two steps.

(3) (*Simulation of the* **find***-stage*) M runs A_1 on input pk, where pk denotes the public key of HIME(R). M also provides A with fair random coins and simulates A's random oracles G and H as follows.

(3.1) When A_1 makes an oracle call $h \in \{0,1\}^{n+k_1}$ of H, M provides A_1 with a random string $H_h \in \{0,1\}^{k_0}$, and adds (h, H_h) to the H-list. M computes $x \in \{0,1\}^{k_0}$ such that $(x + 2^{k_0}h)^2 \equiv y \pmod N$ by using Coppersmith's algorithm if such x exists. Then, M sets $w^* = h||x$.

(3.2) When A_1 makes an oracle call $g \in \{0,1\}^{k_0}$ of G, M provides A with a random string $G_g \in \{0,1\}^{n+k_1}$, and adds (g, G_g) to the G-list.

Let (x_0, x_1, c) be the output with which A_1 halts.

(4) (*Simulation of the* **guess***-stage*) M runs A_2 on input (y, x_0, x_1, c). M responds to oracle queries as follows.

(4.1) Suppose A_2 makes H-query $h \in \{0,1\}^{n+k_1}$. M provides A_1 with a random string $H_h \in \{0,1\}^{k_0}$, and adds (h, H_h) to the H-list. M computes $x \in \{0,1\}^{k_0}$ such that $(x + 2^{k_0}h)^2 \equiv y \pmod N$ by using Coppersmith's algorithm if such x exists. Then, M sets $w^* = h||x$.

(4.2) Suppose A_2 makes G-query $g \in \{0,1\}^{k_0}$. M provides A_2 with a random string $G_g \in \{0,1\}^{n+k_1}$, and adds (g, G_g) to the G-list.

(5) (*Simulation of the decryption oracle*) Suppose A makes a query y' to the decryption oracle. Then, for each (s_i, H_i) that is included in H-list, for each (r_j, G_j) that is included in G-list, machine M

(5.1) Set $t_{i,j} = H_i \oplus r_j$.

(5.2) Compute $x'_{i,j} \in \{0,1\}^n$ and $z'_{i,j} \in \{0,1\}^{k_1}$ such that $x'_{i,j}||z'_{i,j} = s_i \oplus G_j$.

(5.3) Outputs

$$\begin{cases} x'_{i,j} & \text{if it detects an } i,j \text{ such that } z'_{i,j} = 0^{k_1} \text{ and } y' = (s_i||t_{i,j})^2 \bmod N, \\ \text{“Reject”} & \text{otherwise.} \end{cases}$$

(6) M outputs w^* and halts the above simulations if this string was defined in the above process, and **fail** otherwise.

(7) If $w \neq w^*$ and $w + w^* \neq 0 \pmod N$, M computes $\alpha = \gcd(w - w^*, N)$ and outputs a pair of integers that is

$$(p, q) = \begin{cases} (\sqrt[d]{N/\alpha}, \alpha) & \text{if } |\alpha| = \frac{k}{d+1}, \\ (\sqrt[d]{\alpha}, N/\alpha) & \text{otherwise,} \end{cases}$$

and **fail** otherwise.

Remark 4.1. The H-list and G-list include the queries and the correponding answers of both the **find** and **guess** stages of A's execution.

Remark 4.2. When there are plural x in the steps (3.1) and (4.1), M chooses x at random from them, and define w^*.

Remark 4.3. In the step (5), it is not necessary for M to compute $t_{i,j}$, $x'_{i,j}$ and $z'_{i,j}$ whenever the query is submitted to the decryption oracle. M can record them in the table because the number of them are polynomially bounded.

Remark 4.4. In the step (7), notice that if $w \neq w^*$ and $w + w^* \neq 0 \pmod N$ then $p^d \mid w - w^*$ or $q \mid w - w^*$.

We can assume that A never make the same queries to the random oracles in the above process. In (3.1) and (4.1), it is possible to apply Coppersmith's algorithm to compute x, since $2k_0 < k$.

We consider the probability space given by the above process. The inputs N to M are drawn at random according to $\mathcal{G}(1^k)$. We call this "Game 1" and we let $\Pr_1[\cdot]$ denote the corresponding probability.

It is easy to verify that the amount of time t' to carry out Game 1 is

$$t' \leq t + q_H\, T_C(N, 2) + q_G\, q_H\, T_S(k) + \tilde{T}(k) + \mathcal{O}(k).$$

It is also easy to verify that there is a universal machine U such that the computation of M can be done by U^A.

In the step (5), we define the following event to consider the difference between the actual decryption oracle and the simulator of M:

> **FAIL** is true if for a decryption query y' the output of the simulator is different from $\mathcal{D}_{sk}^{G,H}(y')$,

where sk denotes the private key of HIME(R).

Then, we have the following lemma.

Lemma 4.1. The probability that the outputs of the actual decryption oracle and the simulator are different is upper-bounded by

$$\Pr_1[\mathbf{FAIL}] \leq 1 - \left(1 - \frac{2q_G + q_D}{2^{k_0}} - \frac{q_D}{2^{k_1}}\right)\left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D}.$$

Proof. We have

$$\Pr_1[\mathbf{FAIL}] = \Pr_1[\mathbf{FAIL1}] + \Pr_1[\mathbf{FAIL2}],$$

where **FAIL1** and **FAIL2** are events such that

> **FAIL1** is true if for a decryption query y' the decryption oracle outputs a plaintext of y' but M does not.
>
> **FAIL2** is true if for a decryption query M outputs some string (not "Reject") although the decryption oracle outputs "Reject".

FAIL2 occurs only when the decryption fails for the submitted valid decryption query. Namely, for the decryption query y, there are plural i $(1 \leq i \leq 4)$ such that $z_i = 0^{k_1}$ in the decryption procedure of y (cf. Section 3.3). Since the probability that the decryption succeeds for a single valid ciphertext is more than $(1 - 1/2^{k_0} - 1/2^{k_1})^3$ (cf. Theorem 3.1), we have

$$\Pr_1[\mathbf{FAIL2}] \leq 1 - \left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D}. \tag{6}$$

Similarly, we have

$$\Pr_1[\mathbf{FAIL1}] \leq \left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D} \times \Pr_1[\mathbf{FAIL1}^*], \tag{7}$$

where $\mathbf{FAIL1}^*$ is an event such that

FAIL1* is true if for a decryption query y' the decryption oracle outputs a plaintext of y' but M does under the assumption that all valid decryption queries never fail to decrypt.

We discuss $\Pr_1[\mathbf{FAIL1}^*]$. If w^* is defined in the process, then M halts the simulation. Hence, no queries to the decryption oracle are made after w^* is defined. Let y be a target ciphertext, and let y' be the query to the decryption oracle. Let $s_i = x_i \oplus G_{r_i}$, $t_i = r_i \oplus H_{s_i}$, $(s_i||t_i)^2 \equiv y \pmod N$ $(1 \le i \le 4)$, $s' = x'0^{k_1} \oplus G_{r'}$, $t' = r' \oplus H_{s'}$ and $(s'||t')^2 \equiv y' \pmod N$ for $x_i \in \{0,1\}^{n+k_1}$, $x' \in \{0,1\}^n$, $r_i, r' \in \{0,1\}^{k_0}$, $G_{r_i}, G_{r'} \in \{0,1\}^{n+k_1}$, and $H_{s_i}, H_{s'} \in \{0,1\}^{k_0}$. The target ciphertext y includes the information about (s_i, H_{s_i}) and (r_i, G_{r_i}) for $1 \le i \le 4$. And the adversary may utilize the information.

Remark 4.5. There are at most four $X \in \{0,1\}^{k-1}$ such that $X^2 \equiv y \pmod N$ as described in the proof of Theorem 3.1. When the number of such solutions is less than four, we can ignore s_i, t_i, r_i, G_{r_i}, and H_{s_i} for the needless $i \in \{1, \ldots, 4\}$.

We consider the follwing events:

AskR' is true if $(r', G_{r'})$ is on the G-list.

AskS' is true if $(s', H_{s'})$ is on the H-list.

W'=**AskR'** $\wedge$ **AskS'**.

Then, it holds that $\Pr_1[\mathbf{FAIL1}^* \mid \mathbf{W'}] = 0$. Hence, we upper bound $\Pr_1[\mathbf{FAIL1}^*]$ by:

$$\begin{aligned}\Pr_1[\mathbf{FAIL1}^*] &= \Pr_1[\mathbf{FAIL1}^* \mid \mathbf{W'}] \cdot \Pr_1[\mathbf{W'}] + \Pr_1[\mathbf{FAIL1}^* \mid \neg\mathbf{AskR'}] \cdot \Pr_1[\neg\mathbf{AskR'}] \\ &\quad + \Pr_1[\mathbf{FAIL1}^* \mid \mathbf{AskR'} \wedge \neg\mathbf{AskS'}] \cdot \Pr_1[\mathbf{AskR'} \wedge \neg\mathbf{AskS'}] \\ &\le \Pr_1[\mathbf{FAIL1}^* \mid \neg\mathbf{AskR'}] + \Pr_1[\mathbf{AskR'} \wedge \neg\mathbf{AskS'}]. \end{aligned} \tag{8}$$

We first consider $\Pr_1[\mathbf{FAIL1}^* \mid \neg\mathbf{AskR'}]$. We have

$$\begin{aligned}&\Pr_1[\mathbf{FAIL1}^* \mid \neg\mathbf{AskR'}] \\ &= \Pr_1[\mathbf{FAIL1}^* \mid r' \notin \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \cdot \Pr_1[r' \notin \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \\ &\quad + \Pr_1[\mathbf{FAIL1}^* \mid r' \in \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \cdot \Pr_1[r' \in \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \\ &\le \Pr_1[\mathbf{FAIL1}^* \mid r' \notin \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] + \Pr_1[\mathbf{FAIL1}^* \mid r' \in \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}]. \end{aligned} \tag{9}$$

Then it is clear that

$$\Pr_1[\mathbf{FAIL1}^* \mid r' \notin \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \le \frac{q_D}{2^{k_1}}. \tag{10}$$

If $r' = r_i$ for some $1 \le i \le 4$ then it must be $t' = t_i \oplus H_{s_i} \oplus H_{s'}$. However, H-oracle query s_i is not made. Hence the probability that such query y' is made is less than $q_D/2^{k_0}$. It follows that

$$\Pr_1[\mathbf{FAIL1}^* \mid r' \in \{r_1, \ldots, r_4\} \mid \neg\mathbf{AskR'}] \le \frac{q_D}{2^{k_0}}. \tag{11}$$

From (9), (10) and (11), we have

$$\Pr_1[\mathbf{FAIL1}^* \mid \neg\mathbf{AskR'}] \le \frac{q_D}{2^{k_1}} + \frac{q_D}{2^{k_0}}. \tag{12}$$

Next, we consider $\Pr_1[\textbf{AskR'} \wedge \neg\textbf{AskS'}]$. We have

$$\begin{aligned}\Pr_1[\textbf{AskR'} \wedge \neg\textbf{AskS'}] = \Pr_1[s' \notin \{s_1, \ldots, s_4\} \wedge \textbf{AskR'} \wedge \neg\textbf{AskS'}] \\ + \Pr_1[s' \in \{s_1, \ldots, s_4\} \wedge \textbf{AskR'} \wedge \neg\textbf{AskS'}]. \end{aligned} \tag{13}$$

Then, it is clear that

$$\Pr_1[s' \notin \{s_1, \ldots, s_4\} \wedge \textbf{AskR'} \wedge \neg\textbf{AskS'}] \leq \frac{q_G}{2^{k_0}}. \tag{14}$$

If $s' = s_i$ for some $1 \leq i \leq 4$ then it must be $r' \neq r_i$. And any values of H_{s_i} are unknown. Hence, it follows that

$$\Pr_1[s' \in \{s_1, \ldots, s_4\} \wedge \textbf{AskR'} \wedge \neg\textbf{AskS'}] \leq \frac{q_G}{2^{k_0}}. \tag{15}$$

From (13), (14) and (15), we have

$$\Pr_1[\textbf{AskR'} \wedge \neg\textbf{AskS'}] \leq \frac{q_G}{2^{k_0-1}}. \tag{16}$$

From (8), (12) and (16), we have

$$\Pr_1[\textbf{FAIL1}^*] \leq \frac{2q_G + q_D}{2^{k_0}} + \frac{q_D}{2^{k_1}}. \tag{17}$$

We finally obtain

$$\begin{aligned}\Pr_1[\textbf{FAIL}] &\leq \left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D} \times \left(\frac{2q_G + q_D}{2^{k_0}} + \frac{q_D}{2^{k_1}}\right) + 1 - \left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D} \\ &= 1 - \left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D} \left(1 - \frac{2q_G + q_D}{2^{k_0}} - \frac{q_D}{2^{k_1}}\right),\end{aligned}$$

from (6), (7) and (17) □

Intuitively, Lemma 4.1 says that the advantage of M deriving new information from the decryption oracle is negligible. We let $\Pr_2[\cdot] = \Pr_1[\cdot \mid \neg\textbf{FAIL}]$ denote the probability distribution, in Game 1, conditioned on **FAIL** not being true, and call this "Game 2".

Let $s = x_b 0^{k_1} \oplus G_r$ and $t = r \oplus H_s$ in Game 1, where $G_r \in \{0,1\}^{n+k_1}$ and $H_s \in \{0,1\}^{k_0}$. We consider the following event:

BAD is true if :

- G-oracle query r was made in the **find**-stage or the **guess**-stage, and
- $G_r \neq s \oplus x_b 0^{k_1}$

Note that if A makes a H-query h that defines w^*, then M halts. Therefore, a G-query r should be made before (h, H_h) is not on the H-list for such h.

Then we have the following lemma.

Lemma 4.2. The probability that the bad event succeeds is upper-bounded by

$$\Pr_2[\mathbf{BAD}] \ \leq \ \frac{q_G}{2^{k_0}}.$$

Proof. Let consider the following event:

> **AskSr** is true if H-oracle query s was made, and at the point that it was made, r was not on the G-list.

Since $\Pr_2[\mathbf{BAD} \mid \mathbf{AskSr}] = 0$, we then have

$$\begin{aligned}\Pr_2[\mathbf{BAD}] &= \Pr_2[\mathbf{BAD} \mid \mathbf{AskSr}] \cdot \Pr_2[\mathbf{AskSr}] + \Pr_2[\mathbf{BAD} \mid \neg\mathbf{AskSr}] \cdot \Pr_2[\neg\mathbf{AskSr}] \\ &\leq \Pr_2[\mathbf{AskR} \wedge \neg\mathbf{AskSr}] \leq \frac{q_G}{2^{k_0}}.\end{aligned}$$

□

We let $\Pr_3[\cdot] = \Pr_2[\cdot|\neg\mathbf{BAD}]$ denote the probability distribution, in Game 2, conditioned on **BAD** being untrue, and call this "Game 3". Now, we consider the experiment which defines the advantage of A. Namely, choose $N \leftarrow \mathcal{G}(1^k)$ and let $\mathcal{E}_*$ be the corresponding encryption function under HIME(R). Then choose

$$G_*, H_* \leftarrow \mathsf{Hash}; \quad (x_0^*, x_1^*, c^*) \leftarrow A_1^{G_*, H_*, \mathcal{D}_*}(pk); \quad b_* \leftarrow \{0,1\}; \quad y^* \leftarrow \mathcal{E}_*^{G_*, H_*}(x_b^*),$$

and run $A_2^{G_*, H_*, \mathcal{D}_*}(y^*, x_0^*, x_1^*, c^*)$, where pk is a public key of HIME(R), and $\mathcal{D}_*$ is the corresponding decryption function under HIME(R). Let $\Pr_1^*[\cdot]$ be the corresponding distribution and Game 1^* be the game. It is clear that Game 3 and Game 1^* are identical in the sense that the view of A at any point in these two games is the same before the H-query h that defines w^*. Indeed, we have chosen the events **FAIL** and **BAD** so that the oracle queries we are returning in Game 1 will mimic Game 1^* as long as these events remains true.

Let us introduce the following additional events (of Game 3): For $s = x_b 0^{k_1} \oplus G_r$ and $t = r \oplus H_s$,

> **AskH** is true if at the end of the **guess**-stage, (h, H_h) such that $h \in \{0,1\}^{n+k_1}$ and $(h||x)^2 \equiv y \pmod N$ for some $x \in \{0,1\}^{k_0}$ is on the H-list.
>
> **AskR** is true if at the end of the **guess**-stage, (r, G_r) is on the G-list.
>
> **AskS** is true if at the end of the **guess**-stage, (s, H_s) is on the H-list.
>
> $\mathbf{W} = \mathbf{AskR} \wedge \mathbf{AskS}$.

We want to know the relationship between $\Pr_3[\mathbf{AskH}]$ and the advantage of A. However, in Game 3, M haults when w^* is defined. Hence the comparison should be done in Game 1^*. Notice that **AskH**, **AskR**, **AskS** and **W** are the events that are defined before w^* is given and it holds that $\Pr_3[\mathbf{AskH}] = \Pr_1^*[\mathbf{AskH}]$, $\Pr_3[\mathbf{AskR}] = \Pr_1^*[\mathbf{AskR}]$, $\Pr_3[\mathbf{AskS}] = \Pr_1^*[\mathbf{AskS}]$ and $\Pr_3[\mathbf{W}] = \Pr_1^*[\mathbf{W}]$.

Lemma 4.3. The winning probability in Game 3 is bounded below by

$$\Pr_1^*[\mathbf{W}] \geq 2\Pr_1^*[A = b] - 1 - \frac{q_G}{2^{k_0}},$$

where "$A = b$" denotes the event that A is successful in predicting bit b.

Proof. We upper bound $\Pr_1^*[A = b]$ by:

$$\begin{aligned}
\Pr_1^*[A = b] &= \Pr_1^*[A = b \mid \mathbf{W}] \cdot \Pr_1^*[\mathbf{W}] + \Pr_1^*[A = b \mid \neg\mathbf{AskR}] \cdot \Pr_1^*[\neg\mathbf{AskR}] \\
&\quad + \Pr_1^*[A = b \mid \mathbf{AskR} \wedge \neg\mathbf{AskS}] \cdot \Pr_1^*[\mathbf{AskR} \wedge \neg\mathbf{AskS}] \\
&\leq \Pr_1^*[\mathbf{W}] + \Pr_1^*[A = b \mid \neg\mathbf{AskR}] \cdot \Pr_1^*[\neg\mathbf{AskR}] + \Pr_1^*[\mathbf{AskR} \wedge \neg\mathbf{AskS}] \\
&= \Pr_1^*[\mathbf{W}] + \Pr_1^*[A = b \mid \neg\mathbf{AskR}] \cdot (1 - \Pr_1^*[\mathbf{W}] - \Pr_1^*[\mathbf{AskR} \wedge \neg\mathbf{AskS}]) \\
&\quad + \Pr_1^*[\mathbf{AskR} \wedge \neg\mathbf{AskS}] \qquad (18)
\end{aligned}$$

Now observe that if $\neg\mathbf{AskR}$ is true then A has no advantage in predicting b :

$$\Pr_1^*[A = b \mid \neg\mathbf{AskR}] \leq \frac{1}{2}. \qquad (19)$$

We also have

$$\Pr_1^*[\mathbf{AskR} \wedge \neg\mathbf{AskS}] \leq \frac{q_G}{2^{k_0}}. \qquad (20)$$

Therefore, it follows

$$\Pr_1^*[\mathbf{W}] \geq 2\Pr_1^*[A = b] - 1 - \frac{q_G}{2^{k_0}},$$

from (18), (19) and (20). □

From Lemma 4.3 and $\Pr_1^*[\mathbf{W}] = \Pr_3[\mathbf{W}]$, we have

$$\Pr_3[\mathbf{W}] \;\geq\; \epsilon - \frac{q_G}{2^{k_0}} > 0. \qquad (21)$$

Note that $\epsilon - q_G/2^{k_0}$ is non-negligible since ϵ is non-negligible and $q_G/2^{k_0}$ is negligible.

From Lemma 4.1, Lemma 4.2 and (21), we have

$$\begin{aligned}
\Pr_1[\mathbf{AskH}] &\geq \Pr_1[\mathbf{AskS}] \geq \Pr_2[\mathbf{AskS}] \cdot \Pr_1[\mathbf{FAIL}] \\
&\geq \Pr_2[\mathbf{AskS} \mid \neg\mathbf{BAD}] \cdot \Pr_2[\neg\mathbf{BAD}] \cdot \Pr_1[\mathbf{FAIL}] \\
&\geq \Pr_3[\mathbf{AskS}] \cdot \Pr_2[\neg\mathbf{BAD}] \cdot \Pr_1[\mathbf{FAIL}] \\
&\geq \Pr_3[\mathbf{W}] \cdot \Pr_2[\neg\mathbf{BAD}] \cdot \Pr_1[\mathbf{FAIL}] \\
&\geq \left(\epsilon - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{2q_G + q_D}{2^{k_0}} - \frac{q_D}{2^{k_1}}\right)\left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D}.
\end{aligned}$$

The equation $X^2 \equiv y \pmod N$ has four solutions in $\mathbb{Z}_N$, and two of those are less than $N/2$. Since $w, w^* \in \{0,1\}^{k-1}$, the probability that it holds $w \neq w^*$ and $w + w^* \neq 0 \pmod N$ is more than $1/3$. Note that $|N/2| = k - 1$ and that the lowest probability is given when one of four solutions is more than 2^{k-1}. Therefore, we have

$$\begin{aligned}
&\Pr_1[N \leftarrow \mathcal{G}(1^k) \;:\; M(N) = (p, q)] \\
&\qquad \geq \frac{1}{3}\left(\epsilon - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{2q_G + q_D}{2^{k_0}} - \frac{q_D}{2^{k_1}}\right)\left(1 - \frac{1}{2^{k_0}} - \frac{1}{2^{k_1}}\right)^{3q_D}.
\end{aligned}$$

□

4.6 Factoring Problem

In HIME(R), (p, q, d) should be chosen to be intractable to factor $N = p^d q$. Though an efficient algorithm for factoring $N = p^d q$, when d is large ($d \approx \sqrt{\log p}$), is known [7], it is expected that the factoring of N will be intractable when d is small.

Currently known prime factoring algorithms can be divided into two categories, namely those which depend on the size of the composite numbers and those which depend on those prime factors.

As the algorithms depending on prime factors, ρ method, $p-1$ method, $p+1$ method, elliptic curve method([26], [33]) are known. The general number field sieve method depends on the size of the composite number.

HIME(R), in the case of $N = p^2 q$, uses 1024$\sim$1400-bit composite numbers which have 340$\sim$460-bit prime factors and 1400$\sim$1600-bit numbers in the case of $N = p^3 q$ in order to be as strong as 1024-bit RSA-type integers. For these composite numbers, the ρ method is ineffective, and if appropriate primes ($p-1$ and $p+1$ have large prime factors, etc.) are chosen in the key generation of HIME(R), then the $p-1$ and $p+1$ methods are also ineffective.

The elliptic curve method has an amount of calculation for finding a prime factor p of N $L_p[1/2, \sqrt{2}]$ ($L_p[a, b] = \exp((b + o(1))(\log p)^a (\log\log p)^{1-a})$), and an estimation of the amount of calculation for the number field sieve is $L_N[1/3, 1.901]$ ([9]), both of which are subexponential. In practice, depending on the implementation of the algorithm and the ability of computers, the size of the prime factors to be factorized is about $180 \sim 190$-bit by the elliptic curve method, and the size of the composite numbers to be factorized is about 512-bit by the number field sieve method.

More precisely, we estimate the amount of calculation using the above. Let $t_{\mathrm{EC}}(p) = \log(L_p(1/2, \sqrt{(2)}))$ be the logarithm of the amount of calculation by the elliptic curve method and $t_{\mathrm{NFS}}(N) = \log(L_N(1/3, 1.901))$ be that by the number field sieve method.

Then, for 1024-bit RSA-type composite number $n = pq$ (p, q : 512 bits), the number field sieve method is more efficient than elliptic curve method and we have

$$\alpha := t_{\mathrm{NFS}}(\text{1024-bit } N) = C_{\mathrm{NFS}} + 59.42,$$

where C_{NFS} is the constant in the o-factors.

On the other hand, if we use 1344-bit composite number $N = p^2 q$ (or $N = p_1 p_2 p_3$) for HIME(R) (p, q : 448 bits), then the elliptic curve method is more efficient, and we have

$$\beta(448) := \alpha - t_{\mathrm{EC}}(\text{448-bit } p) = C - 0.28,$$

where C is some constant coming from the o-factors. (In Figure 2, we show the graph of β.)

This shows that factoring 1024-bit RSA-type integers is $e^{0.28} = 1.32$ times faster than that of 1344-bit $p^2 q$-type integers. Hence we can say that the modulus of 1344-bit HIME(R) is stronger than that of 1024-bit RSA.

In the case of $N = p^3 q$, HIME(R) uses 1400$\sim$1600 bits N which have 350$\sim$400 bits prime factors. For example, when the bit length of N is 1536, we have

$$\beta(448) = C + 4.9.$$

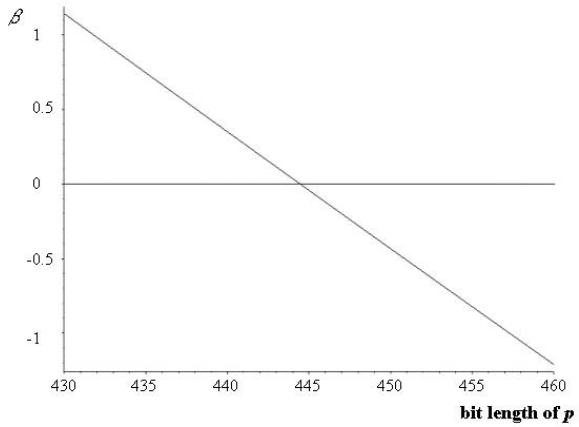

Figure 2: The graph of β

We have $e^{4.9} = 134.3$ and we can say that the strength the integers of this type is comparable with that for 1024-bit RSA-type integers.

Furthermore, we compare the HIME(R)-type integers with 2048-bit and 4096-bit RSA-type integers. If we need the strength such as 2048-bit RSA-type integers, then we use 2304-bit p^2q-type integers or 3072-bit p^3q-type integers, and for 4096-bit RSA-type integers, we use 4032-bit p^2q-type integers or 4928-bit p^3q-type integers (Figure 3). For p^2q-type integers, if the bit length is greater than 2700, then the number field sieve method is more efficient than the elliptic curve method. Thus we can use 4096-bit p^2q-type integers for HIME(R). But we select 4032-bit integers so that the length of p and that of q are same.

On the other hand, the evaluation result based on the implementation is recently reported to compare the integer factoring computational difficulties between $N = pq$ and $N = p^2q$ [12], where p and q are prime numbers. The report guarantees that the computational difficulty of factoring 1024-bit $N = p^2q$ is almost the same with that of factoring 1024-bit $N = pq$.

By the above argument, we recommend the following modulus lengths for HIME(R).

Table 1: Recommendation modulus length for HIME(R)

RSA-type	1024	2048	4096	(bits)
HIME(R) ($N = p^2q$)	$1024 \sim 1344$	2304	4032	(bits)
HIME(R) ($N = p^3q$)	1536	3072	4928	(bits)

4.7 Manger's Attack

Recently, Manger presented the chosen ciphertext attack against PKCS #1 v2.0 [27]. His attack is based on the "integrity check". The actual system must be implemented to be

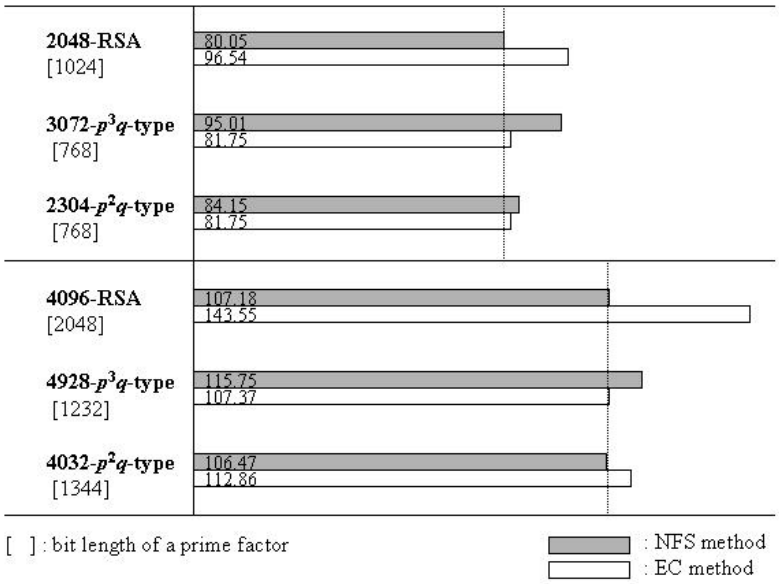

Figure 3: Complexity for long modulus

difficult to distinguish a failure in the integer-to-octets conversion from any subsequent failure, e.g. of the integrity check during OAEP-decoding, to prevent this attack. In this document, we omit the details of countermeasure against this attack, because this problem is not peculiar to HIME(R) but is common to many other public-key cryptosystems.

5 Performance

This section gives the comparison of the basic scheme of HIME(R) and other existing public-key encryption (basic) schemes that are based on the intractability of the factoring problem.

5.1 Key Length

We firstly describe the length of each parameter k_0, k_1 and k in HIME(R). We recommend to take $|k_0|, |k_1| \geq 128$ from security viewpoints.

Next, Table 2 gives the comparison of each modulus length, namely $|k|$, of RSA-OAEP [3], RSA-OAEP+ [36], Rabin-SAEP [6], Rabin-SAEP+[6], EPOC-2 [8], [15] and HIME(R).

Here, it is based on RSA-type 1024, 2048, 4096-bit composite numbers. For p^2q, p^3q-type integers, the lengths recommended with Section 4.6 and lengths which have equal prime factor length to those of RSA-type integers are also made applicable to comparison.

Each modulus length is determined to make the intractability of factoring almost same when NFS and ECM are used (cf. Section 4.6). Here, for simplicity, we omit the bit length for hash functions and parameters that indicate each parameter size.

From Table 3, we can see the key length of HIME(R) is shorter than the other schemes.

5.2 Modular Multiplications

In Table 4, we give the cost for the modular multiplications of the encryption and the decryption in each scheme to evaluate the efficiency of the encryption and the decryption speeds: Here, we consider RSA-OAEP [3], RSA-OAEP+ [36], Rabin-SAEP [6], Rabin-SAEP+[6], EPOC-2 [8, 15] and HIME(R).

In each scheme, the decryption by the Chinese remainder theorem is applied if it can. And, for fairness, the random numbers that are used in all schemes are set 128 bits

We assume that a modular exponentiation a^x (x is k bits) requires $3k/2$ modular multiplications in the standard binary method, and $a^x b^y$ (x, y are k bits) requires $7k/4$ modular multiplications in the extended binary method [23]. And, we set a standard to the number of modular multiplication on 1024 bits modulus. Hence, we assume that an n-bit modular multiplication costs $(n/1024)^2$. The reason for establishing the bit length of each encryption scheme is described in Section 4.6.

In the graphs 4, we give an estimation for bigger modulus.

From these data, we can say that the difference in efficiency among HIME(R) and the above mentioned earlier schemes increases in proportion as $|N|$ increases.

In our scheme, we utilize our calculation method for decryption (cf. Section 2). In the previous method [38], the average total number of modular multiplications T_1 is represented by $T_1 = \frac{|p|}{3} + \frac{25}{6}$, where we make $d = 2$ and set a standard to the number of modular multiplications on N. On the other hand, in our method, the average total number of modular multiplications T_2 is represented by $T_2 = \frac{|p|}{3} + \frac{11}{6}$. Although this difference of efficiency is very small, it is expected that it will be non-negligible in smart card systems and in systems in which match decryption processing must be done at one time. For example, in a system in which 600 times decryptions are done at a single time on average,

Table 2: The length of modulus

	Modulus length (bits)
RSA(-OAEP, OAEP+)	1024
Rabin(-SAEP, -SAEP+)	1024
EPOC-2	$1024 \sim 1152$
HIME(R) ($N = p^2q$)	$1024 \sim 1152$
HIME(R) ($N = p^3q$)	1536
RSA(-OAEP, OAEP+)	2048
Rabin(-SAEP, -SAEP+)	2048
EPOC-2	2304
HIME(R) ($N = p^2q$)	2304
HIME(R) ($N = p^3q$)	3072
RSA(-OAEP, OAEP+)	4096
Rabin(-SAEP, -SAEP+)	4096
EPOC-2	4032
HIME(R) ($N = p^2q$)	4032
HIME(R) ($N = p^3q$)	4928

Table 3: Public and private key length (bits)

	Modulus length	Public key	Secret key
RSA(-OAEP, OAEP+)	1024	1026〜2048	2048
Rabin(-SAEP, SAEP+)	1024	1024	1024
EPOC-2	1024	3072	1024
	1344	4032	1344
HIME(R) ($N = p^2q$)	1024	1024	683
	1344	1344	896
HIME(R) ($N = p^3q$)	(1344	1344	672)
	1536	1536	728
RSA(-OAEP, OAEP+)	2048	2050〜4096	4096
Rabin(-SAEP, SAEP+)	2048	2048	2048
EPOC-2	(2048	6144	2048)
	2304	6912	2304
HIME(R) ($N = p^2q$)	(2048	2048	1366)
	2304	2304	1536
HIME(R) ($N = p^3q$)	(2752	2752	1376)
	3072	3072	1536
RSA(-OAEP, OAEP+)	4096	4098〜8192	8192
Rabin(-SAEP, SAEP+)	4096	4096	4096
EPOC-2	4032	12096	4032
HIME(R) ($N = p^2q$)	4032	4032	2688
HIME(R) ($N = p^3q$)	(4096	4096	2048)
	4928	4928	2464

The data in the parenthesis are for reference.

Table 4: Efficiency by the (converted) number of modular multiplications.

	Modulus length	Encryption	Decryption
RSA(-OAEP, -OAEP+)	1024 (bits)	$2 \sim 1536$	388
Rabin(-SAEP, -SAEP+)	1024 (bits)	1	388
EPOC-2	1024 (bits)	1158	295
	1344 (bits)	2591	667
HIME(R) ($N = p^2q$)	1024 (bits)	1	124
	1344 (bits)	2	275
HIME(R) ($N = p^3q$)	(1344 (bits)	2	146)
	1536 (bits)	3	168
RSA(-OAEP, -OAEP+)	2048 (bits)	$8 \sim 12288$	3088
Rabin(-SAEP, -SAEP+)	2048 (bits)	4	3088
EPOC-2	2048 (bits)	9067	2355
	2304 (bits)	12879	3353
HIME(R) ($N = p^2q$)	(2048 (bits)	4	951)
	2304 (bits)	6	1347
HIME(R) ($N = p^3q$)	(2752 (bits)	8	1246)
	3072 (bits)	9	1320
RSA(-OAEP, -OAEP+)	4096 (bits)	$32 \sim 98304$	24640
Rabin(-SAEP, -SAEP+)	4096 (bits)	16	24640
EPOC-2	4032 (bits)	68466	17957
HIME(R) ($N = p^2q$)	4032 (bits)	16	6974
HIME(R) ($N = p^3q$)	(4096 (bits)	16	4104)
	4928 (bits)	23	5411

The data in the parenthesis are for reference.

Note: In the above table, the orthodox method for exponential calculations is also applied to EPOC-2, because any techniques for speedup were not described in [15]. The developer of EPOC-2, however, says that the efficiency of decryption can be improved by using the speedup technique (e.g., 295 → 180 (1024-bit), 667 → 407 (1344-bit), 2355 → 1444 (2048-bits), 3353 → 2052 (2304-bit)).

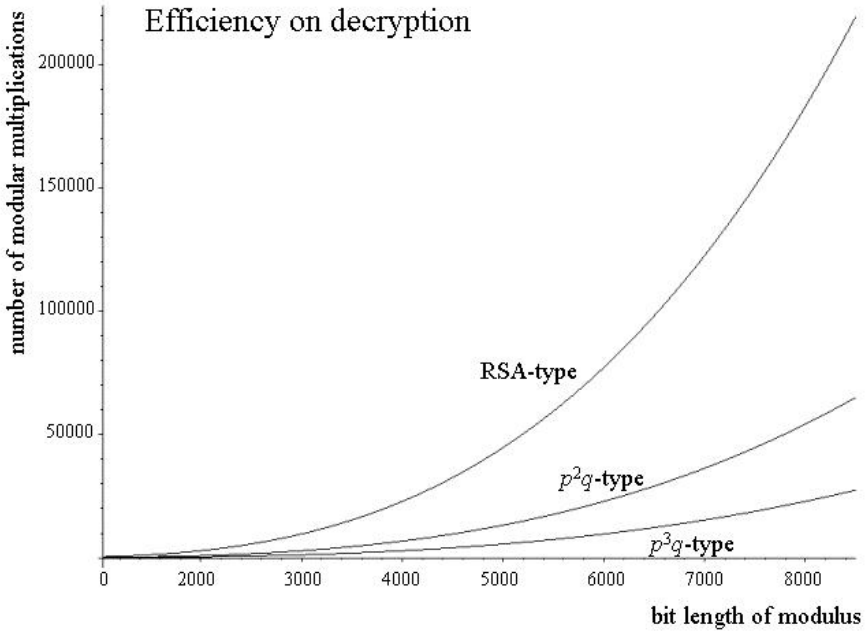

Figure 4: Efficiency on decryption

the difference in the modular multiplications between our method and the previous one amounts to 1400. Furthermore, we believe that the actual decryption speed and mountaing size will be smaller than previous one because ours does not require Euclidean algorithm for Chinese Remainder Theorem.

5.3 Plaintext and Ciphertext Lengths

We show the maximal plaintext and ciphertext lengths, in Table 5. Here, the modular length of each scheme is dominated by Table 2, and the bit length of the random numbers and the check bits are set 128 bits.

The main purpose of the public key cryptosystems is to distribute the data enciphering key of the secret-key cryptosystems. However, to the best of our knowledge, only a few protocols send only the data enciphering key, and many protocols, such as SET, want to send various information (e.g. the identification information of users) with the data encryption key. It is therefore important to choose the public key encryption scheme that reflects the purpose. Hence, this comparison is important for making the purpose of the usage of these public key encryption schemes clear.

The plaintext space of HIME(R) is sufficiently large to send the data encryption key with the attached information.

Table 5: Plaintext and ciphertext lengths (bits)

	Modular length	Plaintext length	Ciphertext length
RSA(-OAEP, -OAEP+)	1024	768	1024
Rabin-SAEP	1024	256	1024
Rabin-SAEP+	1024	384	1024
EPOC-2	1024	Arbitrary	1024+α
	1344	Arbitrary	1344+α
HIME(R) ($N = p^2q$)	1024	768	1024
	1344	1088	1344
HIME(R) ($N = p^3q$)	(1344	1088	1344)
	1536	1280	1536
RSA(-OAEP, -OAEP+)	2048	1792	2048
Rabin-SAEP	2048	512	2048
Rabin-SAEP+	2048	896	2048
EPOC-2	(2048	Arbitrary	2048+α)
	2304	Arbitrary	2304+α
HIME(R) ($N = p^2q$)	(2048	1792	2048)
	2304	2048	2304
HIME(R) ($N = p^3q$)	(2752	2496	2752)
	3072	2816	3072
RSA(-OAEP, -OAEP+)	4096	3840	4096
Rabin-SAEP	4096	1024	4096
Rabin-SAEP+	4096	1920	4096
EPOC-2	4032	Arbitrary	4032+α
HIME(R) ($N = p^2q$)	4032	3776	4032
HIME(R) ($N = p^3q$)	(4096	3840	4096)
	4928	4672	4928

The data in the parenthesis are for reference.

6 Conclusion

This document showed that HIME(R) has almost ideal features as a public-key cryptosystem. HIME(R) is based on a modular squaring over $\mathbb{Z}_N$, where $N = p^d q$ (p and q are prime numbers, $d > 1$), and is secure in the sense of IND-CCA2 under the factoring assumption of N. We showed the performance of HIME(R) by comparing it with that of previous practical factoring base schemes, such as RSA-OAEP or Rabin-SAEP, etc. HIME(R) could gain great profit by taking the modulo $N = p^d q$ instead of $N = pq$. In particular, its decryption speed becomes fast and its difference in efficiency in comparison with the above mentioned earlier schemes increases in proportion as $|N|$ increases (this is owing to the fact that the efficiency of factoring $N = p^d q$ using the elliptic curve method is overtaken by that using the number sieve field method when $|N|$ is large).

We believe that our scheme offers secure and practical public-key encryption that will be useful far into the future.

References

[1] M. Bellare, A.Desai, D.Pointcheval and P. Rogaway. : Relations among notions of security for public-key encryption schemes, *Advances in Cryptology – Crypto'98*, LNCS 1462, Springer-Verlag, pp.26–45 (1998)

[2] M. Bellare and P. Rogaway. : Random oracles are practical – a paradigm for designing efficient protocol, *First ACM Conference on Computer and Communications Security*, pp.62–73 (1993)

[3] M. Bellare and P. Rogaway. : Optimal asymmetric encryption – How to encrypt with RSA, *Advances in Cryptology – Eurocrypt'94*, LNCS 950, Springer-Verlag, pp.92–111 (1994)

[4] D. Bleichenbacher. : Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1, *Advances in Cryptology – Crypto'98*, LNCS 1462, Springer-Verlag, pp.1–12 (1998)

[5] M. Blum and S. Goldwasser. : An efficient probabilistic public-key encryption scheme which hides all partial information, *Advances in Cryptology – Crypto'84*, LNCS 196, Springer-Verlag, pp.289-299 (1985)

[6] D. Boneh. : Simplified OAEP for the RSA and Rabin functions, *Advances in Cryptology – Crypto2001*, LNCS 2139, Springer-Verlag, pp.275-291 (2001)

[7] D. Boneh, G.Durfee and N. Howgrave-Graham. : Factoring $N = p^r q$ for large r, *Advances in Cryptology – Crypto'99*, LNCS 1666, Springer-Verlag, pp.326-337 (1999)

[8] Call for Contributions on New Work Item Proposal on Encryption Algorithms, NTT, 2000-3-10.

[9] D. Coppersmith. : Modifications to the number field sieve, *Journal of in Cryptology*, 6, 3, pp.169-180 (1993)

[10] D. Coppersmith. : Finding a small root of a univariate modular equation, *Advances in Cryptology – Eurocrypt'96*, LNCS 1070, Springer-Verlag, pp.155-165 (1996)

[11] R. Cramer and V. Shoup. : A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack, *Advances in Cryptology – Crypto'98*, LNCS 1462, Springer-Verlag, pp.13-25 (1998)

[12] CRYPTREC Repot 2001, Information-technology Promotion Agency (IPA) (2002), Available from `http://www.ipa.go.jp/security/enc/CRYPTREC/index.html`

[13] D. Dolve, C. Dwork and M. Naor. : Non-malleable cryptography, *Proceedings of the 23rd Annual Symposium on Theory of Computing*, ACM, pp.542–552 (1991)

[14] T. ElGamal. : A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Information Theory*, IT-31, 4, pp.469-472(1985)

[15] EPOC-2 Specification, NTT Information Sharing Platform Laboratories (2001), http://info.isl.ntt.co.jp/epoc/index.html

[16] E. Fujisaki, T. Okamoto and D. Pointcheval : RSA-OAEP is secure under the RSA assumption, *Advances in Cryptology – Crypto2001*, LNCS 2139, Springer-Verlag, pp.269-274 (2001)

[17] D.M. Gordon : Designing and detecting trapdoors for discrete log cryptosystems, *Advances in Cryptology – Crypto'92*, LNCS 740, Springer-Verlag, pp.66-75 (1992)

[18] S. Goldwasser and M. Bellare. : *Lecture Notes on Cryptography* (1997), Available from http:/www-cse.ucsd.edu/users/mihir/

[19] S. Goldwasser and S. Micali: Probabilistic encryption, *Journal of Computer and System Sciences*, 28, 2, pp.270–299 (1984)

[20] Specification of HIME-1 CryptoSystem, Hitachi, Ltd. (2000)

[21] Specification of HIME-2 CryptoSystem, Hitachi, Ltd. (2000)

[22] Specification of HIME(R) CryptoSystem, Hitachi, Ltd. (2001)

[23] D. E. Knuth. : *The Art of Computer Programming*, Addison-Wesley (1981)

[24] N. Koblitz. : Elliptic curve cryptosystems, *Math. Comp.*, 48, 177, pp.203-209 (1987)

[25] A.K. Lenstra and H.W. Lenstra,Jr. : *The Development of the Number Field Sieve*, Lect. Notes Math. 1554, Springer-Verlag (1993)

[26] H.W. Lenstra,Jr. : Factoring integers with elliptic curves, *Annals of Math.*, 126, pp.649-673 (1987)

[27] J. Manger : A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0, *Advances in Cryptology – Crypto2001*, LNCS 2139, Springer-Verlag, pp.230-238 (2001)

[28] A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. : *Handbook of Applied Cryptography*, CRC Press (1996)

[29] V. S. Miller. : Use of elliptic curves in cryptography, *Advances in Cryptology – Crypto'85*, LNCS 218, Springer-Verlag, pp.417-426 (1985)

[30] National Institute of Standards, FIPS Publication 180, Secure Hash Standards (1993)

[31] M.Naor and M.Yung. : Public-key cryptosystems provably secure against chosen ciphertext attacks, *Proceedings of the 22nd Annual Symposium on Theory of Computing*, ACM, pp.427–437 (1990)

[32] M. Nishioka, H. Satoh and K. Sakurai. : Design and analysis of fast provably secure public-key cryptosystems based on a modular squaring, *Proceedings of ICISC2001*, LNCS 2288, Springer-Verlag, pp.81-102 (2001)

[33] J. M. Pollard. : A Monte-Carlo method for factorization, BIT 15, pp.331-334 (1975)

[34] M. O. Rabin. : Digital signatures and public-key encryptions as intractable as factorization, MIT, Technical Report, MIT/LCS/TR-212 (1979)

[35] R. L. Rivest, A. Shamir and L.Adleman. : A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM*, Vol.21, No.2, pp.120-126 (1978)

[36] V. Shoup. : OAEP reconsidered, *Advances in Cryptology – Crypto2001*, LNCS 2139, Springer-Verlag, pp.239-259 (2001)

[37] V. Shoup. : A proposal for an ISO standard for public key encryption (version 2.1), manuscript, Available from `http:/www.shoup.net/`, December 20 (2001)

[38] T. Takagi. : Fast RSA-type Cryptosystem Modulo $p^k q$, *Advances in Cryptology – Crypto'98*, LNCS 1462, Springer-Verlag, pp.318-326 (1998)

[39] H.C.Williams. : A modification of the RSA public key encryption procedure, *IEEE Trans. on Information Theory*, IT-26, 6, pp.726-729 (1980)

[40] H. Woll. : Reductions among number theoretic problems, *Information and Computation*, 72, 3, pp.167-179 (1987)

[41] Y. Zheng and J. Seberry. : Practical approaches to attaining security against adaptive chosen Ciphertext Attacks, *Advances in Cryptology – Crypto'92*, LNCS 740, Springer-Verlag, pp.292-304 (1992)