Evaluation of Security Level of Cryptography: The HIME(R) Encryption Scheme

Alfred Menezes
University of Waterloo
Contact: ajmeneze@uwaterloo.ca

July 31, 2002

Contents

1 Executive Summary

This report evaluates the HIME(R) public-key encryption scheme as specified in [13]. HIME(R) is claimed to be provably IND-CCA2 secure in the random oracle model under the assumption that factoring integers of the form p^2q is intractable. Our conclusions are the following:

1. There are no known risks with using moduli N of the form p^2q. The bitsize of the primes p and q proposed in [13] are appropriate for providing the claimed levels of security.
2. The security proof presented in the HIME(R) self evaluation report [14] is very poorly written and appears to be incomplete. While we expect that the gaps in the proof can be fixed, we feel that significant work must be done in improving the presentation of the proof before it can be accepted as being correct.
3. Assuming that the holes in the security proof for HIME(R) can be filled, we can conclude that the security properties of HIME(R), RSA-OAEP, Rabin-SAEP+ and EPOC-2 are roughly the same. All schemes are provably IND-CCA2 secure in the random oracle model. However, the reduction in the security proofs is tightest for Rabin-SAEP+. RSA-OAEP and Rabin-SAEP+ have the advantage that their security proofs have been more widely scrutinized and the intractability assumptions are more standard.
4. The encryption procedures for HIME(R), RSA-OAEP and Rabin-SAEP+ have approximately the same time complexity, and are all significantly faster than EPOC-2 encryption. HIME(R) decryption is faster than RSA-OAEP, Rabin-SAEP+ and EPOC-2 decryption for larger key sizes (e.g., 4096-bit moduli), but not significantly faster for smaller key sizes (e.g., 1024-bit moduli).

The HIME(R) specification and self evaluation documents are very poorly written. It would not be prudent to accepted HIME(R) as a standard before the HIME(R) documents can be substantially revised. Assuming that the proofs can be fixed, the main advantage of HIME(R) over Rabin-SAEP+ is that the decryption operation in HIME(R) can be several times faster than Rabin-SAEP+ decryption when large moduli are used in high-security applications. A less important advantage is that messages of slightly longer lengths can be encrypted. On the other hand, the security proof in Rabin-SAEP+ is well scrutinized, is very tight, and relies on a more standard intractability assumption (hardness of factoring integers of the form $N = pq$ versus hardness of factoring integers of the form $N = p^2q$). Taking all factors into account, Rabin-SAEP+ appears to have the best security and performance attributes of the four public-key encryption schemes compared in this report.

2 Introduction

HIME(R) is a public-key encryption scheme obtained by combining an extension of the Rabin public-key encryption scheme [24] to moduli N of the form $N = p^d q$, and the Optimal Asymmetric Encryption Padding (OAEP) mechanism of Bellare and Rogaway [3]. The extension of Rabin to moduli $N = p^d q$ is reminiscent of Takagi's extension of RSA to such moduli [29].

The HIME(R) specification document [13] describes the system with modulus $N = p^d q$ for an arbitrary integer $d \geq 2$. However, much of the security and performance analysis is presented only for the cases $d = 2$ and $d = 3$. In addition, page 8 of the specification document [13] states:

> "Although the above algorithm is given for general d, we strongly recommend that $d = 2$ be chosen at present."

We agree with this recommendation because of concerns that integers of the form $p^d q$ are easier to factor for large d (see Section 4.1.1). *Therefore, the remainder of this report shall deal exclusively with the case $d = 2$.*

ORGANIZATION. The remainder of this report is structured as follows. In Section 3, we describe the HIME(R) key generation, encryption, and decryption procedures, and explain why the decryption procedure works. The security of HIME(R) is analyzed in Section 4, where we consider the hardness of the underlying integer factorization problem, and the provable security aspects of HIME(R). In Section 5, we compare the security and performance of HIME(R) to that of the Rabin-SAEP+[1] [5], RSA-OAEP [3] and EPOC-2 [20] public-key encryption schemes. Our conclusions are stated in Section 6.

3 HIME(R) Specification

3.1 HIME(R) Key Pair Generation

To generate a public key pair, an entity A does the following:

1. Select parameters k, n, k_0, k_1 with $k = n + k_0 + k_1 + 1$ and $2k_0 < k$.
2. Select hash function $G : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$ and $H : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$.
3. Select distinct $k/3$-bit primes p and q with $p \equiv q \equiv 3 \pmod 4$.
4. Compute $N = p^2 q$.
5. A's public key is (N, n, k_0, k_1, G, H).

[1] Rabin-SAEP+ is a slight modification of Rabin-SAEP. Rabin-SAEP+ is preferred over Rabin-SAEP because it permits the encryption of longer messages, and the reduction in the security proof is more efficient.

6. A's private key is (p,q).

NOTES.

1. It is expected that in most applications n, k_0, k_1, G and H are *domain parameters*, i.e., common to all entities. In this case, A's public key is comprised only of N.
2. The HIME(R) specification document [13, pages 14-15] presents hash functions G and H for the specific parameters $k = 1344$, $k_0 = 128$, $k_1 = 128$, $n = 1087$. These functions are derived from the SHA-1 hash function [19] in a somewhat complicated manner. Some security rationale should be provided for the choice of this G and H. In particular, it should be explained why G and H were not chosen in the simplest possible way—as the concatenation of SHA-1 values with inputs the seed and a counter.
3. We note the following error in the description of the constants C_i and C in [13, page14]: the subscript 128 (least significant bits) should be a superscript (most significant bits), and the superscript 64 should be a subscript.
4. The HIME(R) specification document [13, page 14] recommends that primes p and q should be *strong primes*. See Section 4.1 for a discussion on strong primes.

3.2 HIME(R) Encryption

To encrypt a message $m \in \{0,1\}^n$ for an entity A with public key N, an entity B does the following:

1. Select $r \in_R \{0,1\}^{k_0}$.
2. Compute $x = \text{OAEP}(m,r) = m0^{k_1} \oplus G(r) \,\|\, r \oplus H(m0^{k_1} \oplus G(r))$ as depicted in Figure 1.
3. Compute $c = x^2 \bmod N$.
4. Output ciphertext c.

NOTES.

1. For the parameter choices $k = 1344$, $k_0 = 128$, $k_1 = 128$, the value of the parameter n should be $n = 1087$, and not $n = 1088$ as given in the HIME(R) specification document [13, page 15]. Ensuring that $n + k_0 + k_1$ is one less than the bitlength k of the modulus N guarantees that $\text{OAEP}(m,r)$ is less than N, and therefore there is no loss of information when the padded message is reduced modulo N.
2. In [13, page 15], the seed r is actually chosen by selecting $R \in_R \{0,1\}^{192}$ and then defining r to be the 128 most significant bits of SHA-1(R). Rationale should be provided on why r is selected in this way.

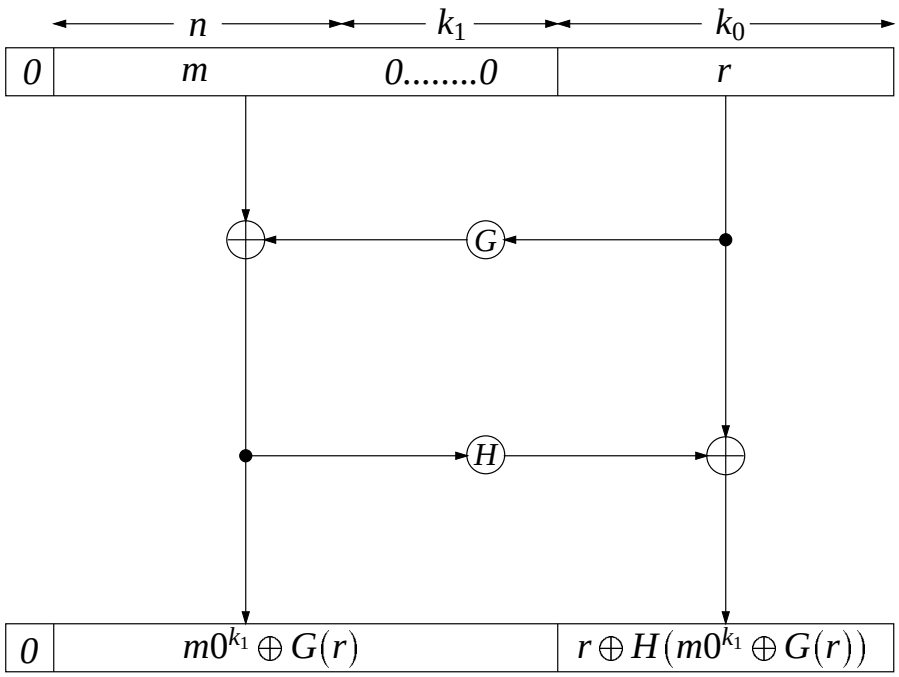

Figure 1: The OAEP encoding function.

3.3 HIME(R) Decryption

To decrypt a ciphertext c, an entity A with private key (p,q) does the following:

1. Compute $z = p^{-1} \bmod q$.
2. Compute $c_p = c \bmod p$ and $c_q = c \bmod q$.
3. Compute $\alpha_1 = c_p^{(p+1)/4} \bmod p$ and $\alpha_2 = p - \alpha_1$.
   If $\alpha_1^2 \not\equiv c_p \pmod{p}$ then return("c is invalid") and stop.
4. Compute $\beta_1 = c_q^{(q+1)/4} \bmod q$ and $\beta_2 = q - \beta_1$.
   If $\beta_1^2 \not\equiv c_q \pmod{q}$ then return("c is invalid") and stop.
5. Compute:
   - 5.1. $u_1 = \alpha_1$, and $v_1 = (\beta_1 - u_1)z \bmod q$.
   - 5.2. $u_2 = \alpha_1$, and $v_2 = (\beta_2 - u_2)z \bmod q$.
   - 5.3. $u_3 = \alpha_2$, and $v_3 = (\beta_1 - u_3)z \bmod q$.
   - 5.4. $u_4 = \alpha_2$, and $v_4 = (\beta_2 - u_4)z \bmod q$.
6. For i from 1 to 4 do:

6.1. Compute $w_i = \left((c-(u_i+v_ip)^2)/pq\right)(2u_i)^{-1} \bmod p$.

6.2. Compute $x_i = u_i + v_ip + w_ipq$.

6.3. Let $x_i = b_i \| s_i \| t_i$, where b_i, s_i, t_i are bit strings of lengths 1, $n+k_1$ and k_0, respectively.

6.4. Compute $r_i = t_i \oplus H(s_i)$ and $M_i = s_i \oplus G(r_i)$.

6.5. Let $M_i = m_i \| p_i$, where m_i and p_i are bit strings of lengths n and k_1, respectively.

6.6. If $b_i = 0$ and $p_i = 0^{k_1}$, then return("plaintext is m_i") and stop.

7. Return("c is invalid").

NOTES.

1. The objective of the decryption procedure is to compute the four square roots x_1, x_2, x_3, x_4 of c modulo N, and then apply the inverse OAEP transform to recover the plaintext. The plaintext that has its first bit b_i equal to 0, and has padding string p_i equal to 0^{k_1} is the one accepted as the valid plaintext. If c does not have any square roots modulo N, or if none of the plaintexts have first bit $b_i = 0$ and padding string $p_i = 0^{k_1}$, then the ciphertext is declared to be invalid.
2. The check that the first bit b_i of a square root is 0 should be included in the decryption procedure in [13, page 16].
3. It is possible that more than one of the plaintexts recovered from x_1, x_2, x_3, x_4 are valid, in which case the decryption procedure may return the wrong plaintext. However, the probability of this event occurring is negligible. This should be noted in the HIME(R) specification document [13].

3.4 Proof that HIME(R) Decryption Works

We provide a brief explanation of the HIME(R) decryption process.

The main goal is to recover the square roots of c modulo N. Now, c is a quadratic residue modulo N if and only if c is a quadratic residue modulo both p and q. Steps 3 and 4 of the decryption process computes the square roots of c modulo p and modulo q; the ciphertext is declared invalid if c is not a quadratic residue modulo N.

Assume now that $\gcd(c,N) = 1$ (which will be the case with overwhelming probability) and that c is a quadratic residue modulo N. Then there are exactly 4 square roots of c modulo N. Each square root x can be written uniquely in the form

$$x = u + vp + wpq,$$

where $u \in [0, p-1]$, $v \in [0, q-1]$, and $w \in [0, p-1]$. Since $x^2 \equiv c \pmod{N}$, we have

$$u^2 \equiv c \pmod{p}$$

and

$$(u+vp)^2 \equiv c \pmod{q}.$$

Thus u is a square root of c modulo p. Also, if β is a square root of c modulo q, then $u+vp \equiv \pm\beta \pmod{q}$ and so

$$v = (\pm\beta - u)p^{-1} \pmod{q}.$$

This explains why u_1, u_2, u_3, u_4 and v_1, v_2, v_3, v_4 are computed in step 5 of the decryption procedure. Finally, we have

$$(u+vp+wpq)^2 \equiv (u+vp)^2 + 2uwpq \equiv c \pmod{N},$$

whence

$$w = \left(\frac{c-(u+vp)^2}{pq}\right)(2u)^{-1} \bmod p.$$

The 4 square roots of c modulo n are denoted $x_i = u_i + v_i p + w_i pq$, $1 \le i \le 4$, in step 6 of the decryption procedure.

4 Security of HIME(R)

4.1 Integer Factorization Problem

HIME(R) bases its security on the assumption that factoring integers N of the form $p^d q$, where p and q are primes of the same bitlength, is intractable.

PROBLEM DEFINITION. The *HIME integer factorization problem* (HIFP) is the following: given a positive integer d and an integer $N = p^d q$ (where p and q are distinct primes of the same bitlength), find p and q. If d is fixed, then we denote the problem by HIFP-d.

Section 4.1.1 provides a summary of the state-of-the-art in algorithms for HIFP. In Section 4.1.2, we explain why there is no need to recommend that the primes p and q be strong primes.

4.1.1 Factoring Integers of the Form $p^d q$

SPECIAL-PURPOSE FACTORING ALGORITHMS. Special-purpose algorithms for HIFP include Pollard's rho algorithm [23], Pollard's $p-1$ algorithm [22], the $p+1$ algorithm [30], and the elliptic curve factoring method (ECM) [16]. Of these, the most powerful is ECM. Its running time

$$O(e^{(\sqrt{2}+o(1))\sqrt{\log p \log\log p}}) \tag{1}$$

depends on the size of the prime factors p of N, so the algorithm tends to find small factors first. The largest prime factor found thus far by ECM is a 55-decimal digit (187-bit) prime factor of the 112-decimal digit (371-bit) number

$$(629^{59}-1)/(628\cdot 3653772966284212495038297 1\cdot 1327481411453869257482884 7)$$

reported by Izumi Miyamoto in October 2001 [18]. (See [11] for the largest numbers factored with ECM.) If both the prime factors p and q of an HIFP modulus N are at least 256 bits in length, then it is extremely unlikely that ECM will be able to factor N.

In 1996, Peralta and Okamoto [21] proposed a modification to the ECM method specially tailored to factor integers of the form $N = p^2q$. However, the extensive analysis and experimentation performed by Ebinger and Teske [10] indicates that the Peralta-Okamoto algorithm is not significantly faster than ECM.

In 1999, Boneh, Durfee and Howgrave-Graham [6] presented an algorithm for factoring $N = p^dq$ using the Lenstra-Lenstra-Lovasz (L^3) lattice basis reduction algorithm. We call their algorithm the *BDH* algorithm. The BDH algorithm runs in *polynomial* (in $\log N$) time if d is large ($d > \log p$). If $d > \sqrt{\log p}$, then the BDH algorithm is faster than the ECM method. If d is constant or small, the running time of the BDH algorithm is *exponential* in $\log N$. In particular, when $N = p^2q$ (i.e., $d = 2$) or $N = p^3q$ (i.e., $d = 3$), then the BDH algorithm is less efficient than the ECM method. The BDH algorithm suggests that factoring $N = p^dq$ becomes easier as d increases (and the bitlength of N remains the same). Therefore, it seems prudent to use moduli $N = p^2q$ instead of $N = p^3q$ in the HIME(R) encryption scheme. In any case, the performance improvements achieved by using p^3q instead of p^2q are not that significant.

GENERAL-PURPOSE FACTORING ALGORITHMS. The number field sieve (NFS) [15] is the most efficient general-purpose factoring algorithm known. It has two main parts: the sieving stage and the matrix stage. The sieving stage, where most of the work is done, can be efficiently parallelized on a network of workstations. In contrast, existing techniques to solve the matrix stage seem to work well only when done on a single large parallel computer. However, recent work by Bernstein [4] suggests that the matrix stage can be performed on a special-purpose machine at a reasonable cost so that the running time is dominated by the time for the sieving stage. The expected running time of the NFS for factoring an integer N is

$$O(e^{(1.923+o(1))(\log N)^{1/3}(\log\log N)^{2/3}}). \tag{2}$$

The NFS was used in 1999 to factor RSA-155, a 155-decimal (512-bit) integer from the RSA Factoring Challenge [9]. Because of the sophisticated nature of the mathematics employed in the NFS and the complicated nature of the algorithm itself, it will be a few years before experts fully understand the capabilities of the NFS for factoring large integers. Indeed, one can reasonably expect that current implementations of NFS can be used to factor numbers of bitsize 600 (or more) within a few months. And this does not take into account improvements in the algorithm itself that are likely to be made in the next year or two (for example, using the ideas in Bernstein's paper [4]). The general conclusion is that 512-bit moduli N are already very

insecure, and that it will be possible for covert efforts (involving only a few people at a single institution, and thus not easily monitored) to factor 768-bit moduli in a few years.

SELECTING PARAMETER SIZES FOR HIME(R). In view of the comments made above, the bitlength t of the primes p and q should be carefully selected so that factoring $N = p^2q$ resists both the ECM and the NFS methods. For example, suppose that one desires a security level that is equal to that of 1024-bit RSA. Then choosing $t = 1024/3 \approx 341$ is potentially problematic because the running time of the ECM[2] is considerably less than the running time of the NFS[3]. Thus, HIME(R) with $t = 341$ (and bitlength $k = 1024$ of N) would be less secure than 1024-bit RSA. If $t = 1344/3 = 448$ (and $k = 1344$), then the running time of the ECM is the same as the running time of the NFS for 1024-bit RSA moduli.

We conclude that the recommendation made in [13, page 17] for using $t = 448$ and $k = 1344$ for HIME(R) moduli $N = p^2q$ in order to achieve the same resistance to ECM as achieved by 1024-bit RSA moduli to NFS is a reasonable one. Similarly, we agree with the recommendation to use $t = 768$ and $k = 2304$ for HIME(R) moduli $N = p^2q$ to achieve the same level of security as 2048-bit RSA, and the recommendation to use $t = 1344$ and $k = 4032$ for HIME(R) moduli $N = p^2q$ to achieve the same level of security as 4096-bit RSA[4].

4.1.2 Strong Primes

The HIME(R) specification document [13, page 14] recommends that the prime factors p, q of the modulus N be strong primes. A prime number p is said to be *strong* if the following conditions hold:

(i) $p+1$ has a large prime factor;

(ii) $p-1$ has a large prime factor denoted r; and

(iii) $r-1$ has a large prime factor.

Strong primes were originally recommended in the late 1970's and early 1980's in order to maximize resistance to the $p-1$ and $p+1$ factoring algorithms (due to Pollard [22] and Williams [30]), and to maximize resistance to the cycling attack of Simmons and Norris [28].

We recall that the $p-1$ and $p+1$ factoring algorithms are "special-purpose" in that they are most effective when $p-1$ or $p+1$ have only small factors. After the discovery of the elliptic curve factoring method (ECM) by Lenstra in 1985 [16], it became clear that the requirements that $p-1$ and $p+1$ each have a large factor are no longer necessary. This is because it is more important to guard against the ECM which is effective whenever an elliptic curve defined over $\mathbb{F}_p$ can be found whose order is smooth—this includes the cases where the elliptic curve has

[2] given by formula (1) without the constants implied by O and $o(1)$.

[3] given by formula (2) without the constants implied by O and $o(1)$.

[4] $k = 4032$ was chosen instead of $k = 4096$ so that $t = k/3$ is an integer multiple of 32.

order $p-1$ or $p+1$. Moreover, Rivest and Silverman [26] have shown that the $p-1$ and $p+1$ factoring algorithms (and also the cycling attacks) are virtually certain to fail if the primes p and q are sufficiently large and chosen at random.

We conclude that while there is no security weakness if p and q are chosen to be strong primes, there is also no reason to require that p and q be strong primes.

4.2 IND-CCA2 Security

The security objective of any public-key encryption scheme is *semantic security against adaptive chosen-ciphertext attacks* [25] (see also [1]). We denote this notion of security by IND-CCA2 (indistinguishability against adaptive chosen-ciphertext attacks). Informally, this means that an adversary can learn nothing about the plaintext corresponding to a given ciphertext c, even when the adversary is allowed to obtain the plaintexts corresponding to ciphertexts (not equal to c) of its choice.

Another equivalent formulation of IND-CCA2 is the following. The goal of an adversary who launches an attack against a legitimate entity is to find two plaintexts for which the adversary is able to distinguish whether a challenge ciphertext is the encryption of the first plaintext or the second plaintext. To achieve this goal, the adversary is permitted to access a decryption oracle that will decrypt any ciphertext of the adversary's choosing, with the exception of the challenge ciphertext. The encryption scheme is IND-CCA2 if no such adversary exists.

HIME(R) bases its security on the hardness of HIFP-2, i.e., the problem of factoring $N = p^2q$ where p and q are primes of the same bitlength. The *HIFP-2 assumption* is that there is no efficient algorithm for solving the HIFP-2 problem.

4.2.1 Problems with the Security Proof

The HIME(R) evaluation report [14] includes a proof that the HIME(R) encryption scheme is IND-CCA2 secure in the random oracle model under the HIFP-2 assumption. In the random oracle model, the hash functions G and H are modeled by random functions.

The security proof is very poorly written and it is difficult to read the proof and conclude with certainty that it is correct.

The following are some minor comments on the proof of Theorem 2.2 in [14].

1. The simulators of the G and H oracles are not correctly described. The simulator should first check if a query was made previously in which case the previous answer should be returned.

2. More explanation is needed on the derivation of the running time t' of the simulator. At first glance, it seemed to me that the term '$q_G q_H T_S(k)$' should in fact be '$q_D q_G q_H T_S(k)$'

since for each decryption query the simulator of the decryption oracle has to search through all pairs from the Cartesian product of the H-list and the G-list. However, on further reflection, I now believe that the searches can be scheduled so that no pair is ever examined twice and hence the expression for t' is correct. In any case, more explanation would help remove any doubts a reader may have.

3. The statement of Lemma 2.1 in [14] is a little confusing. What do the authors mean when they talk about the difference between the "outputs of the actual decryption oracle" and the "outputs of the simulator"? In indistinguishability proofs, we want to show that the simulator could "simulate" the read "decryption oracle" successfully. That is, the HIME(R)-attacker should not be able to distinguish the simulator from a real decryption oracle until the simulator terminates the simulation.

A more serious concern with the proof is the following. Let w^* denote a desired square root of the target ciphertext y^*. Then w^* implicitly defines the values s^*, t^* and r^*. It is clear that the HIME(R)-attacker $A = (A_1, A_2)$ does not have any advantage in winning its game if it does not learn the value of $H(s^*)$. If A queries the H-oracle with s^*, then the simulator M wins its game and succeeds (with high probability) in factoring N. *However, there is another way in which A can learn the value of $H(s^*)$, namely through some decryption oracle query. It appears that this possibility has not been considered in the security proof.*

Let R_1 denote the event that A queries the H-oracle with s^*, and let R_2 denote the event that A learns the value of $H(s^*)$ through some decryption oracle query. Let ε be A's advantage. Then we can show that $\Pr(R_1) + \Pr(R_2) \geq \varepsilon$. By not considering the possibility of event R_2, the authors have implicitly assumed that $\Pr(R_1) \geq \varepsilon$.

This gap must be fixed to complete the proof. It is possible that the event R_2 occurs with negligible probability, in which case the proof is complete—but the authors have to show this. On the other hand, it this cannot be shown, then the event R_2 may have to be addressed in a similar way that Boneh handled the analogous situation in his security proof of Rabin-SAEP [5].

This reviewer will be pleased to review a substantially revised version of the proof and modify his assessment if the above comments can be properly addressed.

4.2.2 Random Oracle Assumption

It is worth repeating that as with the known security proofs of Rabin-SAEP+, RSA-OAEP and EPOC-2, the security proof for HIME(R) takes place in the random oracle model [2], and therefore does not imply security in the real world where the hash functions G and H are no longer random functions. This perspective was given some credence by Canetti, Goldreich and Halevi [8] who provided examples of signature and encryption schemes which they proved secure in the random oracle model, but which are insecure with *any* reasonable instantiation of

the random function. However, the encryption and signature schemes in [8] are rather contrived, and a security proof in the random oracle model is now widely accepted as providing valuable *heuristic* evidence for security. Additionally, a security proof of HIME(R) in the random oracle model implies that any attack on HIMR(R) must either solve the HIFP-2 problem, or exploit some properties of the hash functions employed.

5 Comparisons

We briefly compare the HIME(R), Rabin-SAEP+ [5], RSA-OAEP [3, 27, 12] and EPOC-2 [20] public-key encryption schemes.

5.1 Provable Security

All four public-key encryption schemes have been proven IND-CCA2 secure in the random oracle model where the hash functions employed are modeled as random functions. Moreover, none of the schemes have been proven secure under a standard intractability assumption only. Thus, all four schemes are subject to the same criticisms regarding the limitations of interpreting security proofs in the random oracle model in the real world (see Section 4.2.2).

We note that the security proofs of RSA-OAEP and Rabin-SAEP+ have undergone greater public scrutiny that the proofs of HIME(R) and EPOC-2. This is an important point considering how complicated security proofs are for public-key encryption schemes, and how subtle flaws in proofs can go unnoticed for many years.[5]

The intractability assumptions for HIME(R) and EPOC-2 are the same—hardness of factoring integers of the form $N = p^2q$. The intractability assumption for Rabin-SAEP+ is hardness of factoring integers of the form $N = pq$. The intractability assumption for RSA-OAEP is hardness of finding e^{th} roots modulo N, where $N = pq$. As discussed in Section 4.1.1, factoring integers of the form $N = p^2q$ is conceivably easier than factoring integers $N = pq$, although nothing has been proven in support of this. It is also conceivable that finding e^{th} roots modulo N is easier than factoring N—some limited theoretical evidence in support of this statement is provided by Boneh and Venkatesan [7] for the case of small e (e.g., $e = 3$). Thus one can conclude that the intractability assumption for Rabin-SAEP+ is more believable than the intractability assumptions for HIME(R), EPOC-2, and RSA-OAEP. It should be noted, however, that no unconditional result has been proven on the relative difficulty of these problems.

[5] A notable example of this is the 1994 Bellare-Rogaway security proof of RSA-OAEP [3] that was found to be flawed by Shoup [27] in 2000.

5.2 Tightness of the Reduction

Security proofs typically work by providing a *reduction* from the solution of a problem P to the breaking of a cryptographic scheme S. That is, one shows how P can be solved if one is given a hypothetical oracle for breaking S. In the case of the HIME(R) security proof, S is HIME(R) and P is the HIFP-2 problem. The success of the reduction is measured by the tightness of the reduction. More precisely, a reduction is *tight* if when an adversary can break S in time at most t and with probability at least ε, then the reduction algorithm that solves P runs in time at most t' with success probability at least ε', where $t' \approx t$ and $\varepsilon' \approx \varepsilon$. A tight reduction is important because one then has the assurance that the security level of S is very closely related to the security level of P. Thus one can select security parameters in S to be of the same size as the parameters that make P intractable against all known attacks. For example, if the reduction in a security proof of HIME(R) is tight, then one can use 1344-bit moduli N and be assured that breaking HIME(R) will require roughly the same amount of work as it takes to factor N.

We let q_D, q_G and q_H denote the maximum number of queries an IND-CCA2 adversary of HIME(R) is allowed to make. Such an adversary is said to $(t, q_D, q_G, q_H, \varepsilon)$-break HIME(R) if its running time is at most t steps and it meets its objective with advantage at least ε. Let us assume that the gaps in the HIME(R) security proof can be filled without affecting the running time of the reduction or the probability of success. Theorem 2.2 of [14] states that if an IND-CCA2 adversary succeeds in $(t, q_D, q_G, q_H, \varepsilon)$-breaking HIME(R), then the reduction algorithm described in the proof of Theorem 2.2 solves the HIFE-2 problem in time

$$t' = t + q_H T_C(N, 2) + q_G q_H T_S(k) + \text{poly}(k) \tag{3}$$

with success probability

$$\varepsilon' \geq \frac{1}{3}\left(\varepsilon - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_D}{2^{k_1-1}} - \frac{q_D}{2^{k_0}}\right), \tag{4}$$

where $T_C(N, 2)$ is the time to apply Coppersmith's algorithm, and $T_S(k)$ is the time to encrypt a message. For a 1344-bit modulus N, the suggested value for parameters k_0 and k_1 is 128. For concreteness, we let $q_G \approx 2^{60}$, $q_H \approx 2^{60}$ and $q_D \approx 2^{30}$. Then from (3) and (4) we see that

$$t' \approx t + q_G q_H \tag{5}$$

and

$$\varepsilon' \approx \varepsilon/3. \tag{6}$$

Since $q_G q_H \approx 2^{120}$, the reduction is *not* tight.

Therefore, in order to be assured from the HIME(R) security proof that HIME(R) provides a certain acceptable level of security, one may have to use a modulus N that is significantly bigger than one would actually use in practice. Thus, it is not clear what assurances are actually provided by the HIME(R) security proof.

We note that the security proof for Rabin-SAEP+ in [5] is very tight ($t' \approx t + q_D + q_H + q_G$ and $\varepsilon' \approx \varepsilon/6$). The security proofs for RSA-OAEP in [12] and for EPOC-2 in [20] are not as tight as the Rabin-SAEP+ proof, but roughly as tight as the HIME(R) proof.

5.3 Efficiency

We suppose that the bitlength k of the modulus $N = p^2q$ for HIME(R) and EPOC-2 and the bitlength K of the modulus $N = pq$ for RSA-OAEP and Rabin-SAEP+ have been selected so that the security level of these moduli against integer factorization attacks is the same. The bitlength of the prime factors of a HIME(R) or EPOC-2 modulus is denoted by t (so $t \approx k/3$), while the bitlength of the prime factors of an RSA-OAEP or Rabin-SAEP+ modulus is denoted by T (so $T \approx K/2$).

We focus our discussion on the parameter sets $(k,t,K,T) = (1344, 448, 1024, 512)$ and $(k,t,K,T) = (2304, 768, 2048, 1024)$. For these parameter sets, $k > K$ and $t < T$.

Recall that the key pairs for the 4 encryption schemes are the following:

1. *HIME(R)*: Public key is N; private key is (p,q).
2. *RSA-OAEP*: Public key is N; private key is (p,q,d). (We assume that the encryption exponent is $e = 3$.)
3. *Rabin-SAEP+*: Public key is N; private key is (p,q). (We assume that the encryption exponent is $e = 2$.)
4. *EPOC-2*: Public key is (N,g,h); private key is (p,q,L). Here, $g,h \in [1, N-1]$, and $L \in [1, p-1]$.

5.3.1 Encryption

The computational steps that dominate the execution time of the encryption process for the 4 encryption schemes are:

1. *HIME(R)*: $m^2 \bmod N$. That is, a modular squaring operation with a k-bit modulus.
2. *RSA-OAEP*: $m^3 \bmod N$. That is, a modular squaring operation and a modular multiplication operation modulo a K-bit modulus.
3. *Rabin-SAEP+*: $m^2 \bmod N$. That is, a modular squaring operation with a K-bit modulus.
4. *EPOC-2*: $g^r h^e \bmod N$, where r and e are t-bit and $2t$-bit integers, respectively, whose values depend on the message being encrypted. That is one modular exponentiation with a k-bit modulus and a t-bit exponent, and another modular exponentiation with a k-bit modulus and a $2t$-bit exponent.

Rabin-SAEP+ encryption is slightly faster than HIME(R) encryption since the modulus for the former is smaller than the modulus for the latter. And, HIME(R) encryption can be expected to be faster than RSA-OAEP despite the smaller modulus for RSA-OAEP since the latter requires 2 modular operatations. However, it is fair to say that the running times are sufficiently close to each other that it is doubtful that an application would favour one encryption scheme over the other purely on the basis of encryption speeds.

EPOC-2 encryption can be sped up by using "Shamir's trick" for simultaneous exponentiation (Algorithm 14.88 in [17]). Moreover, one could precompute powers of g and h to further speed up the exponentiations. However, even with these enhancements, EPOC-2 encryption is expected to be significantly slower than HIME(R), RSA-OAEP and Rabin-SAEP+ encryption.

5.3.2 Decryption

The computational steps that dominate the execution time of the decryption process for the 4 encryption schemes are:

1. *HIME(R)*: $c^{(p+1)/4} \bmod p$ and $c^{(q+1)/4} \bmod q$. That is, two modular exponentiations with t-bit moduli.
2. *RSA-OAEP*: $c^{d_p} \bmod p$ and $c^{d_q} \bmod q$. That is, two modular exponentiations with T-bit moduli.
3. *Rabin-SAEP+*: $c^{(p+1)/4} \bmod p$ and $c^{(q+1)/4} \bmod q$. That is, two modular exponentiations with T-bit moduli.
4. *EPOC-2*: $c^{p-1} \bmod p^2$ and $g^r h^e \bmod q$. That is, one modular exponentiation with a $2t$-bit modulus and an t-bit exponent, and two modular exponentiations with an t-bit modulus and an t-bit exponent[6].

RSA-OAEP and Rabin-SAEP+ decryption have the same time complexity. Since $t < T$, HIME(R) decryption is expected to be faster than RSA-OAEP and Rabin-SAEP+ decryption. For smaller k and K (e.g. $(k,K) = (1344,1024)$) it is fair to say that the running times are sufficiently close to each other that it is doubtful that an application would favour one decryption scheme over the other purely on the basis of decryption speeds[7]. However for larger k and K (e.g., $(k,K) = (4032,4096)$), t will be significantly smaller than T, and hence HIME(R) decryption can be expected to be several times faster than RSA-OAEP and Rabin-SAEP+ decryption.

EPOC-2 decryption will be slower than HIME(R), RSA-OAEP and Rabin-SAEP+ decryption. However, the difference in speed will not be major since the computation $g^r h^e \bmod q$ will take

[6] Note that the $2t$-bit exponent e can be first reduced modulo $q-1$.

[7] The footnote on page 23 of [14] reports a running time of 17ms for a HIME(R) decryption and 22 ms for an RSA-OAEP decryption. The moduli sizes k and K were not given, but presumably $(k,K) = (1344,1024)$

about the same time as an t-bit exponentiation in HIME(R) if powers of g and h are precomputed. The reason for the slower decryption in EPOC-2 is the larger $2t$-bit modulus in the exponentiation operation $c^{p-1} \bmod p^2$.

6 Conclusions

HIME(R) PARAMETERS.

1. Security rationale should be provided for the choices of the hash functions G and H.
2. The suggested parameters $k_0 = 128$, $k_1 = 128$, G and H are appropriate for the parameter $k = 1344$. Recommendations for k_0, k_1, G and H should also be provided for the parameters $k = 2304$ and $k = 4032$.
3. Guidance should be provided on selecting parameters k, k_0, k_1, G and H when HIME(R) is used to transport symmetric keys of bitlengths 80, 128, 192 and 256.

FACTORIZATION OF THE MODULUS $N = p^2q$.

1. We agree with the recommendation that moduli $N = p^2q$ be used with HIME(R), and not moduli $N = p^dq$ with $d \geq 3$.
2. If the recommended bitlengths t are used for the primes p and q ($t = 448$, $t = 768$ and $t = 1344$ for security equivalent to 1024-bit RSA, 2048-bit RSA, and 4096-bit RSA, respectively), then there are no integer factorization algorithms known that will decrease the claimed security level of these moduli.
3. There is no reason to mandate the use of strong primes in HIME(R).

SECURITY PROOF THAT HIME(R) IS IND-CCA2 SECURE. The security proof presented in the HIME(R) self evaluation report [14] is very poorly written and appears to be incomplete. Since HIME(R) is simply the Rabin encryption scheme with moduli of the form $N = p^2q$ combined with the OAEP formatting mechanism, we expect that one should be able to prove HIME(R) secure by closely following the security proof of Rabin-SAEP+ in [5]. We therefore expect that the gaps in the proof can be fixed, but feel that significant work must be done in improving the presentation of the proof before it can be accepted as being correct.

COMPARISONS WITH RABIN-SAEP+, RSA-OAEP AND EPOC-2.

1. Assuming that the holes in the security proof for HIME(R) can be filled, we can conclude that the security properties of HIME(R), RSA-OAEP, Rabin-SAEP+ and EPOC-2 are roughly the same. All schemes are provably IND-CCA2 secure in the random oracle model. However, the reduction in the security proofs is the tightest for Rabin-SAEP+.

RSA-OAEP and Rabin-SAEP+ have the advantage that their security proofs have been more widely scrutinized and the intractability assumptions are more standard.

2. The encryption procedures for HIME(R), RSA-OAEP and Rabin-SAEP+ have approximately the same time complexity, and are all significantly faster than EPOC-2 encryption. HIME(R) decryption is faster than RSA-0AEP, Rabin-SAEP+ and EPOC-2 decryption for larger key sizes (e.g., 4096-bit moduli), but not significantly faster for smaller key sizes (e.g., 1024-bit moduli).

SUMMARY. The HIME(R) specification and self evaluation documents are very poorly written. In particular, the security proof appears to be incomplete. It would not be prudent to accepted HIME(R) as a standard before the HIME(R) documents can be substantially revised. This reviewer will be pleased to review substantially revised versions and modify his assessment if the shortcomings noted can be properly addressed.

Assuming that the proofs can be fixed, the main advantage of HIME(R) over Rabin-SAEP+ is that the decryption operation in HIME(R) can be several times faster than Rabin-SAEP+ decryption when large moduli are used in high-security applications. A less important advantage is that messages of slightly longer lengths can be encrypted. On the other hand, the security proof in Rabin-SAEP+ is well scrutinized, is very tight, and relies on a more standard intractability assumption (hardness of factoring integers of the form $N = pq$ versus hardness of factoring integers of the form $N = p^2q$). Taking all factors into account, Rabin-SAEP+ appears to have the best security and performance attributes of the four public-key encryption schemes compared in this report.

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*, Lecture Notes in Computer Science, **1462** (1998), Springer-Verlag, 26-45.

[2] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security*, 1993, 62-73. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html.

[3] M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology—Eurocrypt '94*, Lecture Notes in Computer Science, **950** (1995), Springer-Verlag, 92-111. Full version available at http://www.cs.ucdavis.edu/~rogaway/papers/index.html.

[4] D. Bernstein, "Circuits for integer factorization: A proposal", November 2001. Available at http://cr.yp.to/papers.html.

[5] D. Boneh, "Simplified OAEP for the RSA and Rabin functions", *Advances in Cryptology—Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 275-291.

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

[7] D. Boneh and R. Venkatesan, "Breaking RSA may not be equivalent to factoring", *Advances in Cryptology—Eurocrypt '98*, Lecture Notes in Computer Science, **1403** (1998), Springer-Verlag, 59-71.

[8] R. Canetti, O. Goldreich and S. Halevi, "The random oracle methodology, revisited", *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, 1998, 209-218.

[9] S. Cavallar et al., "Factorization of a 512-bit RSA modulus", *Advances in Cryptology—Eurocrypt 2000*, Lecture Notes in Computer Science, **1807** (2000), Springer-Verlag, 1-18.

[10] P. Ebinger and E. Teske, "Factoring $N = pq^2$ with the elliptic curve method", Technical Report CORR 2002-02, 2002. Available at http://www.cacr.math.uwaterloo.ca/tech_reports.html. Final version to appear in *Proceedings of ANTS-V*.

[11] The ECMNET Project, http://www.loria.fr/~zimmerma/records/ecmnet.html.

[12] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, "RSA-OAEP is secure under the RSA assumption", *Advances in Cryptology—Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 260-274.

[13] Hitachi Ltd., “Specification of HIME(R) cryptosystem”, 2002.

[14] Hitachi Ltd., “Self evaluation report: HIME(R) cryptosystem”, 2002.

[15] A. Lenstra and H. Lenstra, *The Development of the Number Field Sieve*, Springer-Verlag, 1993.

[16] H. Lenstra, “Factoring integers with elliptic curves”, *Annals of Mathematics*, **126** (1987), 649-673.

[17] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[18] I. Miyamoto, “A report on integer factorization by elliptic curve method”, personal communication, October 2001. Available at http://www.loria.fr/~zimmerma/records/p55b.

[19] National Institute of Standards and Technology, *Secure Hash Standard (SHS)*, FIPS Publication 180-1, 1995.

[20] NTT Information Sharing Platform Laboratories, “EPOC-2 specification”, September 26 2001.

[21] R. Peralta and E. Okamoto, “Faster factoring of integers of a special form”, *IEICE Trans. Fundamentals, E79-A*, **4** (1996), 489-493.

[22] J. Pollard, “Theorems on factorization and primality testing”, *Proceedings of the Cambridge Philosophical Society*, **76** (1974), 521-528.

[23] J. Pollard, “Monte Carlo methods for index computation”, *BIT*, **15** (1975), 331-334.

[24] M. Rabin, ‘Digitalized signatures and public-key functions as intractable as factorization”, MIT Lab. for Computer Science, Technical Report LCS/TR-212, 1979.

[25] C. Rackoff and D. Simon, “Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack”, *Advances in Cryptology—Crypto ’91*, Lecture Notes in Computer Science, **576** (1992), Springer-Verlag, 433-444.

[26] R. Rivest and R. Silverman, “Are ‘strong’ primes needed for RSA?”, Report 2001/007, *Cryptology ePrint Archive*, 2001. Available at http://http://eprint.iacr.org/

[27] V. Shoup, “OAEP reconsidered”, *Advances in Cryptology—Crypto 2001*, Lecture Notes in Computer Science, **2139** (2001), Springer-Verlag, 239-259. Full version available at http://shoup.net/papers/.

[28] G. Simmons and M. Norris, “Preliminary comments on the M.I.T. public-key cryptosystem”, *Cryptologia*, **1** (1977), 406-414.

[29] T. Takagi, "Fast RSA-type cryptosystem modulo $p^k q$", *Advances in Cryptology—Crypto '98*, Lecture Notes in Computer Science, **1462** (1998), Springer-Verlag, 318-326.

[30] H. Williams, "A $p+1$ method of factoring", *Mathematics of Computation*, **39** (1982), 225-234.