Evaluation Report of HIME(R)

July 19th, 2002

Abstract

In this report we evaluate the HIME(R), which is the public-key cryptosystem developed by HITACHI, Ltd. In section 1 we estimate the security level of the key size for the composite numbers p^2q and p^3q that are recommended in the specification of HIME(R). In section 2 we evaluate the correctness of the security proof described in the self-evaluation of HIME(R). In section 3 we compare the security and the plaintext size of HIME(R), RSA-OAEP, RSA-OAEP+, Rabin-SAEP, and Rabin-SAEP+. In section 4 we estimate the comparison between HIME(R) primitive, RSA primitive, Rabin primitive, and RSA-type primitive modulo p^dq on their efficiency.

1 Selecting the Size of the Modulus p^dq

HIME is an asymmetric crypto-system of RSA-type, which uses composite moduli of the form $N = p^dq$. In particular, for HIME it is proposed $d = 2$ or $d = 3$. The security proof of HIME is based on the assumption, that it is intractable to factor such a modulus (that is, the probability to factor such a modulus efficiently is negligible). We comment on the according part of the self-evaluation report (section 2.6) given by Hitachi Ltd.

The goal is to estimate the size for a HIME-modulus N such that we expect the same computational work to factoring N as to factoring an RSA-modulus N' of certain bit size. Specifically, the following claim is made:

Claim 1: *Let $N = p^dq$ denote a HIME-modulus, where p and q are of about equal size, e.g. let $|size(p) - size(q)| \leq 1$. Let $N' = p'q'$ denote a RSA-modulus, where p and q are of about equal size, e.g. let $|size(p') - size(q')| \leq 1$. Then*

- *if $d = 2$ then the expected computational work to factor the HIME-modulus N of size 1344, 2304, and 4032 is about as much as the expected computational work to factor the RSA-modulus N' of size 1024, 2048, and 4096 bits, respectively;*

- *if $d = 3$ then the expected computational work to factor the HIME-modulus N of size 1536, 3072, and 4928 is about as much as the expected computational work to factor the RSA-modulus N' of size 1024, 2048, and 4096 bits, respectively.*

In order to justify the claim, two types of factoring algorithms have been taken into account: General factoring algorithms with an expected running time only dependent of the modulus N itself, and special factoring algorithms with a running time dependent of the largest prime factor P of the modulus N. The asymptotically fastest algorithms from these families are the Generalized Number Field Sieve (GNFS) and the Elliptic Curve Method (ECM). It has been pointed out that the special algorithm from [BDG99] is not efficient for $d = 2$ or $d = 3$. We define

$$L_x[\epsilon, c] = \exp((\ln x)^\epsilon (\ln \ln x)^{1-\epsilon}) \tag{1}$$

for all real $x > e \approx 2.71828\ldots$ and all real constants ϵ, c with $0 \leq \epsilon \leq 1$ and $c > 0$. (In the self-evaluation report L is defined by $L_x[\epsilon, c] = \exp((c+o(1))((\ln x)^\epsilon (\ln \ln x)^{1-\epsilon}))$, but it is more convenient to move the term $o(1)$ out of the definition of L.) The GNFS has an expected asymptotic running time of order of magnitude

$$L_N[\frac{1}{3}, \sqrt[3]{64/9} + o(1)] \ , \tag{2}$$

where $\sqrt[3]{64/9} \approx 1.923$. By a theoretical modification of the GNFS due to Coppersmith it gets the slightly better expected asymptotic running time of order of magnitude

$$L_N[\frac{1}{3}, \sqrt[3]{(92 + 26\sqrt{13})/27} + o(1)] \ , \tag{3}$$

where $\sqrt[3]{(92 + 26\sqrt{13})/27} \approx 1.902$. In the self-evaluation report the asymptotics 3 is assumed for the GNFS. We note that the authors of [RSA155] and [LV01] assumed the asymptotics 2 for the GNFS, thus we recommend to use the asymptotics 2 here, too.

The ECM has an expected asymptotic running time of order of magnitude

$$L_P[\frac{1}{2}, \sqrt{2} + o(1)] \ . \tag{4}$$

We note that the asymptotics 2 (or 3) and 4 haven't rigorously proven, yet. However, the assumptions of these asymptotics are accepted in general.

We also note that the ECM can be sped up for numbers of the form $N = p^2 q$, see [PO96], this speed-up hasn't been taken into account in the self-evaluation report (See the recent experimental result [ET02]). If the exponent d is large, then two algorithms are known for factoring the special form $N = p^d q$ in polynomial time [BDG99] [CUS02].

For all practical considerations and extrapolations the term $o(1)$ has been omitted (as is customary), although if an extrapolation far out is made, one cannot neglect that $o(1) \to 0$, thus extrapolated running times will be an overestimate.

The current factoring records are: RSA-155, a 512-bit composite, has been split by the GNFS in 2000, where approximately 8000 MIPS-years have been spent [RSA155]; a 55-digit factor, that is, a 183-bit factor, has been found by the ECM in 2001 (see `http://www.loria.fr/~zimmerma/records/ecmnet.html`). Thus, if $N = p^d q$, then N has to be much larger than 2^{512}, and p and q have to be much larger than 2^{183}.

It must be noted that ECM can be run in parallel without much communication overhead, thus an ECM factorization could be (and have already been) distributed over a network. On the other side, the GNFS works in two stages; the first stage (the sieving-stage) can also be run in parallel without much communication overhead and thus is likewise suitable for distributed computation, while the second stage (the linear-algebra-stage) has to be performed on a single node and requires exceptional amounts of main memory. Therefore, a compound effort to find a factor by the ECM using more unused resources that are available to the Internet is likely to find larger factors (say 60 to 65 digits), merely using current computer technology.

We take the asymptotics 2 or even 3 for the GNFS and 4 for the ECM for granted. The working hypothesis of the self-evaluation report is as follows:

Hypothesis 1 *Let d be a small integer and $N = \prod_{0 \le i \le d} p_i$ be a composite, where the $d+1$ prime factors p_i are of about the same size, e.g. let $|size(p_i) - size(p_j)| \le 1$ for all $0 \le i, j \le d$. Then there exists an integer N_0 such that for all composites N as above, $N \ge N_0$, the* expected *running time of the ECM exceeds the* expected *running time of the GNFS.*

The hypothesis is justified by the asymptotics for the GNFS and the ECM with N given as in the hypothesis. For instance, the GNFS factors a 1024-bit RSA-modulus faster than the ECM.

To estimate the size of N, $N = p^2 q$ or $N = p^3 q$, such that factoring N by the ECM is about as difficult as factoring an RSA-modulus N' of given size, the authors of the self-evaluation report compared the asymptotics 3 and 4 for both algorithms based on the special form of N as in the hypothesis. For convenience, the following functions have been defined:

$$t_{EC} = \ln\Big(L_P[\frac{1}{2}, \sqrt{2}]\Big) = \sqrt{2 \ln P \ln \ln P}$$

and

$$t_{NFS} = \ln\Big(L_N[\frac{1}{3}, 1.901]\Big) = 1.901 \sqrt[3]{\ln N (\ln \ln N)^2}$$

The estimation of the size of the HIME-moduli is demonstrated by two examples:

1. Let N' be a 1024-bit RSA-modulus. How large must a HIME-modulus $N = p^2q$ be, such that the expected computational work to factor both with the best known algorithms is about equal?

   To do this, equations 3 and 4 have been equated, finally neglecting any O-constants. For example, the authors define $\alpha = t_{NFS}(1024 - \text{bit}N) = C_{NFS} + 59.42$, while $t_{NFS}(1024 - \text{bit}N) \approx 59.48$, thus $C_{NFS} \approx 0$. There's no comment indicating why the constant is or can be neglected.

   By means of trial or Newton-approximation (this hasn't been made clear), one gets that$t_{NFS}(1024 - \text{bit}N) - t_{EC}(448 - \text{bit}P) = C - 0.28$ for a constant C that is implicitly neglected, and the authors concluded that the expected computational work to factor N' with bit size 1024 by the GNFS is about as much as the expected computational work to factor N with bit size $3 \cdot 488 = 1344$ (with a factor of $e^{0.28} \approx 1.32$, all other factors have been dropped).

2. A similar calculation for a HIME-modulus $N = p^3q$ leads to $size(N) = 1536$ when $size(N') = 1024$ for RSA-moduli N'.

This shall justify the initial claim, the other presented estimates.

Additionally we notice that Silverman estimated the key length of the Multi-Prime RSA, PKCS #1 [PKCS], which uses the modulus $n = p_1p_2...p_k$ where $p_1, p_2, ...p_k$ are prime numbers with the same size [Sil00]. He concluded that breaking a 1024-bit RSA modulus is as hard as the 1024-bit modulus of the Multi-Prime RSA with 3 primes.

1.1 Conclusion

The estimation of the key sizes has been done with some bias and sloppiness. Our objections were:

1. For the GNFS the asymptotics 3 has been used, while in recent research papers (e.g. [LV01] or [RSA155]) one finds the asymptotics 2. Thus, the authors assumed an algorithm for factoring RSA-moduli that is too fast.

2. The speed-up of the ECM for special numbers of the form $N = p^2q$ as described in [PO96] [ET02] should be taken into account for the estimate.

3. The authors naively equated 3 and 4 without taking into account any data points (such as 8000 MIPS-years spent for RSA-155).

4. Many assumptions made aren't stated explicitly, these should be clarified. Likewise, the overall model is not clearly explained.

Although the estimates are not grossly wrong, we would appreciate if the authors would make their estimates with more care and precision. That is, the authors should describe their model more precisely; they should not make implicit assumptions, and if the make them explicit, the authors should justify them clearly. The authors should collect data points for ECM in order to equate the expected computational work for GNFS and ECM.

2 Proof of Security

HIME is an asymmetric crypto-system of RSA-type, which uses composite moduli of the form $N = p^d q$. In particular, for HIME it is proposed $d = 2$ or $d = 3$. It is proved that if HIME can be broken efficiently, then N can be factored efficiently with non-negligible probability. Assuming the intractability of the factoring problem, HIME cannot be broken efficiently with non-negligible probability.

We comment on the according part of the self-evaluation report given by Hitachi Ltd. We mainly evaluate the correctness of the proof of Theorem 2.2 in the self-evaluation report.

We shortly review the encryption function of HIME(R). Let n be a k-bit public modulus of HIME(R) and let G, H be two hash functions, where $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0-1}, H : \{0,1\}^{k-k_0-1} \to \{0,1\}^{k_0}$ for integers k_0, k_1 such that $2k_0 < k, n = k - k_0 - k_1 - 1 > 0$. In the proof of HIME(R), the hash fucntion is considered as the random oracle. The padding scheme of the HIME(R) for a n-bit message m is computed as follow:

$$x = s||(r \oplus H(r)), \quad s = m0^{k_1} \oplus G(r). \tag{5}$$

Then $y = x^2 \bmod n$ is the ciphertext of the message m. This padding scheme of HIME(R) is based on the OAEP conversion [BR94].

The techniques used in the security proof are similar with that of SAEP/SAEP+ [Bon01]. In the security proof of HIME(R), the simulator tries to find the non-trivial square root using the Coppersmith algorithm. The authors of HIME(R) proved that the success probability of the simulator becomes non-negligible in the random oracle model using the adversary that breaks the semantically security against the adaptive chosen ciphertext attack. A similar proof idea was discussed for the OAEP-RSA with low encryption exponent $e = 3$ in the manuscript [Sho01b].

Because the security proof of HIME(R) follows from these well-studied standard techniques, we have not noticed technical flaws in their proof. However, we have found several editorial mistakes in the proof of Theorem 2.2. We list up these mistakes in the following.

p. 10–11 (simulation of the find- and guess-stages)

There is a common omission in (3.1), (3.2), (4.1), and (4.2): Whenever h or g can be found on the respective list (i.e. a previous oracle call is repeated), the previously defined value H_h or G_g must be returned.

p. 11 (simulation of the decryption oracle)

The presentation is confusing: While steps (5.1) and (5.2) are performed for each (s_i, H_i) and each (r_j, G_j), step (5.3) is not.

Also note that for consistency with the specification of the actual decryption procedure, the decryption oracle should return "reject" rather than $*$ in the second branch of (5.3).

p. 14 (additional events of Game 1)

Note that event AskH (and thus, as a special case, also AskS) is impossible in Game 1. According to (3.1) and (4.1) in the definition of the simulator (pp. 10–11), no h such that $(h \parallel x)^2 \equiv y \pmod{N}$ for some $x \in \{0,1\}^{k_0}$ will ever be put on the H-list; instead the simulation will be aborted when such a h is encountered. (Also cf. p. 11: "M does not return [to A] the answer of h which defines w^*".)

HITACHI must change the definition of H-list. The goal of the simulation in the proof is to find h as the query to random oracle H, which satisfies $(h||x)^e = y \bmod n$ for some integer x and a given y. If H-list does not include such h, event AskH is empty. Thus they have to add the h in the first component of H-list before simulator M terminates.

p. 15 (inequality (9))

There appears to be a typo in inequality (9): "$\Pr_2[\mathbf{W}]$" is probably intended to read "$\Pr_3[\mathbf{W}]$".

It is better to explain the meaning of the value $\epsilon - \frac{q_G}{2^{k_0}}$, which is a positive non-negligible function because ϵ is non-negligible and $\frac{q_G}{2^{k_0}}$ is negligible.

2.1 Conclusion

We evaluated the security of HIME(R) based on the self-evaluation report, namely the correctness of the proof of Theorem 2.2. There are several editorial mistakes in the proof of Theorem 2.2, but the general outline of their proof is correct.

3 Comparison of HIME with other Crypto-Systems

In this section the security and the plaintext size of HIME(R) are compared with other cryptosystems. We compare HIME(R) with the RSA-OAEP [BR94], RSA-OAEP+ [Sho01a], Rabin-SAEP [Bon01], and Rabin-SAEP+ [Bon01].

3.1 Comparison of Security Reduction

Let A denote an attacker that breaks one of the following crypto-systems (i.e. IND-CCA2) in time t with probability ε. Then there exists an algorithm A' that factors the modulus N (pq for RSA and Rabin, pq^d, $d = 2, 3$ for HIME) in time t' with probability ε'. T_C denotes the running time of Coppersmith's algorithm, T_s and $\tilde{T}$ are as in the HIME report.

	Running time t' of A'	Success-Probability ε' of A'
HIME	$t + q_H T_C(N, 2) + q_G q_H T_s(k) + \tilde{T}(k) + O(k)$	$\frac{1}{3}\left(\varepsilon - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_G}{2^{k_0}}\right)\left(1 - \frac{q_D}{2^{k_1}} - \frac{q_D}{2^{k_0}}\right)$
RSA-OAEP	$2t + q_H(q_H + 2q_G)O(k^3)$	$\frac{\varepsilon^2}{4} - \varepsilon\left(\frac{2q_D q_G + q_D + q_G}{2^{k_0}} + \frac{2q_D}{2^{k_1}} + \frac{32}{2^{k-2k_0}}\right)$
RSA-OAEP$^+$	$O(t + q_G q_H T_f + (q_G + q_{H'} + q_H + q_D)k)$	$\varepsilon - \frac{q_{H'} + q_D}{2^{k_1}} - \frac{(q_D+1)q_G}{2^{k_0}}$
Rabin-SAEP	$t + O(q_H + q_G + q_D)$	$\varepsilon\left(1 - \frac{q_D}{2^{s_0}} - \frac{q_D}{2^{s_1}}\right)$
Rabin-SAEP$^+$	$t + O(q_D q_H T_C(n, 2) + q_D T_C(n, 4))$	$\frac{1}{6}\varepsilon\left(1 - \frac{2q_D}{2^{s_0}} - \frac{2q_D}{2^{s_1}}\right)$

We assume that the numbers of queries q_H, q_G, and q_D are same. The running time of the security reduction of the Rabin-SAEP is the linear order of q, where q is one of the q_H, q_G, and q_D. The running times of the security reduction of other cryptosystems have the quadratic order of q. Therefore, the running time of the security reduction of HIME is as fast as those of RSA-OAEP, Rabin-SAEP, and Rabin-SAEP+.

The security parameters k_0, k_1 of HIME and s_0, s_1 are chosen enough large. Then the probabilities of security reduction of HIME, RSA-OAEP+, Rabin-SAEP, Rabin-SAEP+ are the order of $\mathcal{O}(\varepsilon)$, and that of RSA-OAEP is the order of $\mathcal{O}(\varepsilon^2)$.

Therefore we conclude the security reduction of HIME is similar as that of the Rabin-SAEP+.

3.2 Comparison of Plaintext Lengths

They summarize the size of the plaintext and ciphertext in Table 5. The main advantage of the HIME cryptosystem is the size of the plaintext. The size of the plaintext of 1344-bit HIME is 1088 bits, which is equal to or the largest among other RSA/Rabin-type cryptosystems.

4 Comparison of Efficiency as Primitives

In this section we discuss the estimation of the efficiency in the self-evaluation report of HIME. We discuss the estimation about the HIME cryptosystem, the RSA cryptosystem, the Rabin cryptosystem, the RSA-type cryptosystem modulo $p^k q$ [Tak98] (We call it PkQ cryptosystem). In this document, we estimate the efficiency of the decryption of the 1024-bit RSA/Rabin cryptosystem and the 1344-bit HIME/PkQ cryptosystem.

Let $Primitive(sk, C)$ be the cryptographic primitives of these cryptosystems, where sk is a secret key and C is a ciphertext. Let $Check(M)$ be the checking mechanism using the random hash functions of these cryptosystems in order to achieve the IND-CCA security. The whole decryption algorithm of these cryptosystems first computes $M = Primitive(sk, C)$ for a given ciphertext C using the secret key sk, and then check the correctness of the ciphertext by $Check(M)$. Because the computation of $Check(M)$ is very fast, we will estimate the primitive part of the decryption algorithms in the following.

In the specification form (page 7), they state "*The actual decryption speed mountaing size will be smaller than previous one because ours does not require Euclidean algorithm for CRT*". However, this is misleading, although they insist that this is one of the advantage of the HIME.

The standard RSA decryption using the CRT, which is described in the PKCS #1 [PKCS], also does not require Euclidean algorithm, because the Garner's algorithm is used for the CRT. If we apply the Garner's algorithm, the other RSA/Rabin-type cryptosystems do not require an inversion operation for the CRT. We describe the decryption algorithm of the RSA cryptosystem with CRT from PKCS #1 in the following.

```
INPUT p,q, dp, dq, p_inv_q, C
OUTPUT M
1:  Mp = mod_pow(C,dp,p);
2:  Mq = mod_pow(C,dq,q);
3:  V = (Mq - Mp)*(p_inv_q) mod q; <--- pre-comp. 1/p mod q
5:  Return M = Mp + p*V;
```

Algorithm 1: PKCS #1 Decryption

Here p and q are primes with same bit-length and the RSA modulus is pq. $d_p = d \bmod p-1$, $d_q = d \bmod q-1$, and $p\_inv\_q = p^{-1} \bmod q$ are the secret key of the PKCS #1. C is a ciphertext of the message M. $\mathsf{mod\_pow}(a, b, c)$ means $a^b \bmod c$.

Note that there is no inversion during whole decryption process of the decryption of PKCS #1. The overhead of the decryption algorithm over the computation of the

modular multiplications $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$ is just 1 modular multiplication of k bits and 1 multiplication of k bits and k bits, where k is the bit-length of prime p (or q). They are at most 0.5 modular multiplication of $2k$ bits. Therefore, if we estimate the efficiency based on section 3.2 of the self-evaluation report of the HIME(R), the decryption time of the PKCS #1 for 1024-bit modulus becomes 385 modular multiplications of 1024 bits. The estimations in Table 3 in section 3.2 of the self-evaluation report must be corrected.

On the contrary, HIME(R) always has at least one inversion during the decryption process for p^2q. An inversion operation $c^{-1} \bmod p$ is estimated about 30 times slower in software than the modular multiplication $c_1c_2 \bmod p$, where c, c_1, c_2 are as large as p [OS01]. The coprocessor of the inversion is not equipped on many smartcards. Therefore, the inversion is critical operation on them and the estimation must carefully consider it. They do not state how the computation time of inversion is estimated in the self-evaluation report of the HIME(R).

Takagi proposed an RSA-type cryptosystem using the modulus p^dq [Tak98] [Tak01]. We call it the PkQ cryptosystem in the following. The decryption of the HIME cryptosystem uses a similar technique with the PkQ cryptosystem. In section 3.2 of the self-evaluation report of HIME, they estimate the decryption time of the HIME as $\frac{|p|}{3} + \frac{11}{3}$ in comparison with that of the PkQ as $\frac{|p|}{3} + \frac{25}{6}$. Moreover, they also stated "*It has less modular multiplications than previous one*" in section 2 of the specification of HIME. However, these statements are misleading. We discuss the estimations are not correct in the following subsections. We conclude the decryption of the PkQ cryptosystem is faster than that of the HIME. Our estimation of the overheads is 17.7 modular multiplications for the HIME decryption and 14.8 modular multiplications for the PkQ decryption for the 1344-bit modulus. Thus the total decryption times are 275 for the HIME decryption and 272 for the PkQ decryption, respectively.

For the comparison of the HIME decryption and the PkQ decryption we use the standard algorithms described in book [MvOV97]. A multiplication of n bits and t bits requires $(n+1)(t+1)$ single-precision multiplications (Algorithm 14.12 and Note 14.15 of [MvOV97]). A division of n bits by t bits requires $(n-t)(t+3)$ single-precision multiplications (Algorithm 14.20 and Note 14.25 of [MvOV97]). Let $a^b \bmod c$ be a modular multiplication of n bits, where a, b, and c are n-bit integers. The modular multiplication of n bits is $2n^2 + 5n + 1$ single-precision multiplications (Algorithm 14.28 of [MvOV97]). We assume the computation time of an addition and a subtraction is negligible, comparing with the multiplication or the division.

4.1 Performance of the Rabin based Decryption

We describe the decryption algorithm of the Rabin based decryption modulo p^2q in the following. We use the secret key p_inv_q in addition to p and q as the PKCS # does.

```
INPUT p, q, p_inv_q, C
OUTPUT M
0:  Check C mod p and C mod q are quadratic residue;
1:  Mp = mod_pow(C,(p+1)/4,p);
2:  Mq = mod_pow(C,(q+1)/4,q);
3:  V = (+-Mq - (+-Mp))*(p_inv_q) mod q; <--- pre-comp. 1/p mod q
4:  M = +-Mp + p*V;
5:  Find the proper M from 4 ambiguities
6:  Return M;
```

Algorithm 2: Rabin based decryption

Assume that the value $p^{-1} \bmod q$ is pre-computed. Let k be the bit-length of prime p (or q). We estimate the computation efficiency of step 3 and step 4, which are the overhead parts from the two modular exponentiations $C^{(p+1)/4} \bmod p$ and $C^{(q+1)/4} \bmod q$. There are 4 different solutions and we estimate the efficiency in the case of finding all the four solutions.

We conclude that the overhead of the Rabin based decryption is at most 1.5 modular multiplications of $2k$ bits. Therefore, if we estimate the efficiency based on section 3.2 of the self-evaluation report, the decryption time of the Rabin based decryption becomes 386 modular multiplications of 1024 bits with pre-computation of $p^{-1} \bmod q$. The details of the estimation is as follows:

Theorem 1 *Fro step 3 and step 4 of the HIME decryption (algorithm 3), we need (1)2 modular multiplications of k bits, (2)4 multiplications of k bits and k bits, where k is the bit-length of the prime p of q. If the parameter $p^{-1} \bmod q$ is not pre-computed, we need (1)2 modular multiplications of k bits, (2)4 multiplications of k bits and k bits, and (3)1 modular inverse of k bits.*

Proof: In step 3 we compute four values $V = (\pm M_q - (\pm M_p)) * (p\_inv\_q) \bmod q$ and we assign them as $V_{0,0} = (M_q - M_p) * (p\_inv\_q) \bmod q$, $V_{1,0} = (M_q + M_p) * (p\_inv\_q) \bmod q$, $V_{0,1} = -V_{1,0} \bmod q$, and $V_{1,1} = -V_{0,0} \bmod q$. We need 2 modular multiplications of k bits. If $p^{-1} \bmod q$ is not pre-computed, we need one more modular inverse of k bits.

In step 4 we compute four values $M = \pm Mp + p * V$ and we assign them as $M_{0,0} = M_p + p * V_{0,0}$, $M_{1,0} = -M_p + p * V_{1,0}$, $M_{0,1} = M_p + p * V_{0,1}$, and $M_{1,1} = -M_p + p * V_{0,0}$. We need 4 multiplications of k bits. ■

4.2 Performance of the HIME Decryption

We describe the decryption algorithm of the HIME modulo for the modulus p^2q in the following. In section 2.2.1 of the specification of the HIME(R), the secret key

is only the primes p and q. In section 3.3, the secret keys are not only p and q but also $(p+1)/4$, $(q+1)/4$, and $p^{-1} \bmod q$. We do not understand which secret keys are the proper specification of the HIME(R). If the parameters $p^{-1} \bmod q$ and pq are pre-computed, the decryption speed of the HIME(R) can be improved.

```
INPUT p, q, p_inv_q, pq, C
OUTPUT M
0:  Check C mod p and C mod q are quadratic residue;
1:  Mp = mod_pow(C,(p+1)/4,p);
2:  Mq = mod_pow(C,(q+1)/4,q);
3:    r0 = +-Mp mod p;
4:    r1 = (+-Mq - r0)*(p_inv_q) mod q; <--- pre-comp. 1/p mod q
5:      R1 = r0 + p*r1 ;
6:      E  = C - R1^2 mod p^2q;
7:      F  = E/pq;
8:      r2 = F*(2r0)^{-1} mod p;        <--- inversion modulo p.
9:  M = R1 + (pq)*r2;
10: Find the proper M from 4 ambiguities
11: Return M;
```

Algorithm 3: HIME decryption

Assume that the values $p^{-1} \bmod q$ and pq are pre-computed as secret keys. Let k be the bit-length of prime p (or q). We estimate the computation efficiency from step 3 to step 9 of the HIME decryption, which is the overhead part from the two modular exponentiations $C^{(p+1)/4} \bmod p$ and $C^{(q+1)/4} \bmod q$. There are 4 different solutions and we estimate the efficiency in the case of finding all the four solutions.

From step 3 to step 9 of the HIME decryption (algorithm 3), we need $64k^2+100k+20$ single-precision multiplications and 2 modular inverses of k bits, where k is the bit-length of the prime p (or q) (See theorem 2). We estimate in the case of $k = 448$, which is the 1344-bit HIME cryptosystem. A modular multiplication of 1024 bits is $2 * (1024)^2 + 5 * (1024) + 1 = 2,102,273$ single-precision multiplications. The overhead of the HIME decryption without the two modular inverses is $64 * (448)^2 + 100 * 448 + 20 = 12,889,876$ single-precision multiplications. A modular inverse of k bits is 30 times as fast as a modular multiplication [OS01]. Thus we estimate the modular inverse of k bits requires $30 * (2 * (448)^2 + 5 * 448 + 1) = 12,109,470$ single-precision multiplications. Thus total overhead is $37,108,816$ single-precision multiplications. We conclude that the overhead of the HIME decryption is 17.7 modular multiplications of 1024 bits with the pre-computations of $p^{-1} \bmod q$ and pq. The details of the estimation is as follows:

Theorem 2 *From step 3 to step 9 of the HIME decryption (algorithm 3), we need (1)*$64k^2 + 100k + 20$ *single-precision multiplications and (2)2 modular inverses of k bits, where k is the bit-length of the prime p of q. If the parameters* $p^{-1} \bmod q$ *and*

pq are not pre-computed, we need (1)$65k^2+102k+21$ *single-precision multiplications and (2)3 modular inverses of k bits.*

Proof: We refer the decryption algorithm in section 3.7 of the specification of the HIME.

In step 3 we compute two values $r_0 = \pm M_p \bmod p$, where we assign them as $r_{0,0} = M_p \bmod p$ and $r_{0,1} = -M_p \bmod p$. This computation is negligible.

In step 4 we compute four values $r_1 = (\pm M_q - r_0) * (p^{-1}) \bmod q$ and we assign them as $r_{1,00} = (M_q - r_{0,0}) * (p^{-1}) \bmod q$, $r_{1,01} = (M_q - r_{0,1}) * (p^{-1}) \bmod q$, $r_{1,10} = (-M_q - r_{0,0}) * (p^{-1}) \bmod q$, and $r_{1,11} = (-M_q - r_{0,1}) * (p^{-1}) \bmod q$. We need 4 modular multiplications of k bits, which is $8k^2 + 20k + 4$ single-precision multiplications. If $p^{-1} \bmod q$ is not pre-computed, we need an additional modular inverse of k bits.

In step 5 we compute four values $R_1 = r_0 + p * r_1$ and we assign them as $R_{0,0} = r_{0,0} + p * r_{1,00}$, $R_{0,1} = r_{0,1} + p * r_{1,01}$, $R_{1,0} = r_{0,0} + p * r_{1,10}$, and $R_{1,1} = r_{0,1} + p * r_{1,11}$. We need 4 multiplications of k bits and k bits, which is $4k^2 + 8k + 4$ single-precision multiplications.

In step 6 we compute $E = C - R_1^2 \bmod p^2q$ and we assign them as $E_{0,0} = C - R_{0,0}^2 \bmod p^2q$, $E_{0,1} = C - R_{0,1}^2 \bmod p^2q$, $E_{1,0} = C - R_{1,0}^2 \bmod p^2q$, and $E_{1,1} = C - R_{1,1}^2 \bmod p^2q$. The bit length of $R_{i,j}$ for $i, j = 0, 1$ is $2k$. We need 4 multiplications of $2k$ bits and $2k$ bits, and 4 division of $4k$ bits by $3k$ bits, which is $28k^2 + 28k + 4$ single-precision multiplications.

In step 7 we compute $F = E/(pq)$ and we assign them as $F_{0,0} = E_{0,0}/(pq)$, $F_{0,1} = E_{0,1}/(pq)$, $F_{1,0} = E_{1,0}/(pq)$, and $F_{1,1} = E_{0,1}/(pq)$. We need 4 divisions of $3k$ bits by $2k$ bits, which is $8k^2 + 12k$ single-precision multiplications.

In step 8, we compute $r_2 = F * (2r_0)^{-1} \bmod p$ and we assign them as $r_{2,00} = F_{0,0} * (2r_{0,0})^{-1} \bmod p$, $r_{2,01} = F_{0,1} * (2r_{0,1})^{-1} \bmod p$, $r_{2,10} = F_{1,0} * (2r_{0,0})^{-1} \bmod p$, and $r_{2,11} = F_{1,1} * (2r_{0,1})^{-1} \bmod p$. The value r_0 is not same for each decryption and we cannot pre-compute the value $(2r_0)^{-1} \bmod p$. Therefore, we need 2 modular inversions of k bits, and 4 modular multiplications of k bits, which is $8k^2 + 20k + 4$ single-precision multiplications.

Finally, in step 9, we compute $M = R_1 + (pq) * r_2$ and we assign $M_{00} = R_{0,0} + (pq) * r_{2,00}$ $M_{01} = R_{0,1} + (pq) * r_{2,01}$ $M_{10} = R_{1,0} + (pq) * r_{2,10}$, and $M_{11} = R_{1,1} + (pq) * r_{2,11}$. We need 4 multiplications of k bits and $2k$ bits, which is $8k^2 + 12k + 4$ single-precision multiplications. ■

4.3 Performance of the PkQ Decryption

We describe the fast decryption algorithm of the Rabin version of the PkQ cryptosystem. The secret keys of the PkQ cryptosystem are $p, q, (p^2)^{-1} \bmod q$, and p^2. The general description is as follows:

```
INPUT p, q, p2_inv_q, p^2, C
OUTPUT M
0:  Check C mod p and C mod q are quadratic residue;
1:  Mp = mod_pow(C,(p+1)/4,p);
2:  Mq = mod_pow(C,(q+1)/4,q);
3:    F = (+-Mp)^2 mod p^2;
4:    E = C - F mod p^2;
5:    B = E/p;
6:    K = (+-Mp)*B*(2*F)^{-1} mod p; <--- inversion modulo p
7:    A = +-Mp + p*K;
8:  V = (+-Mq - A)*(p2_inv_q) mod q; <--- pre-comp. 1/p^2 mod q
9:  M = +-Mp + (p^2)*V;
10: Find the proper M from 4 ambiguities
11: Return M;
```

Algorithm 4: PkQ decryption

Assume that the values $p^{-1} \bmod q$ and p^2 are pre-computed as secret keys. Let k be the bit-length of prime p or q. We estimate the computation efficiency from step 3 to step 9, which is the overhead part from the two modular exponentiations $C^{(p+1)/4} \bmod p$ and $C^{(q+1)/4} \bmod q$. There are 4 different solutions and we estimate the efficiency in the case of finding all the four solutions.

From step 3 to step 9 of the PkQ decryption (algorithm 4), we need $48k^2 + 88k + 16$ single-precision multiplications and 2 modular inverse of k bits, where k is the bit-length of the prime p of q (See theorem 3). We estimate in the case of $k = 448$, which is the 1344-bit PkQ cryptosystem. A modular multiplication of 1024 bits is $2*(1024)^2+5*(1024)+1 = 2,102,273$ single-precision multiplications. The overhead of the PkQ decryption without two modular multiplications is $34*(448)^2+72*448+16 = 6,856,208$ single-precision multiplications. A modular inverse of k bits is 30 times as fast as a modular multiplication [OS01]. Thus we estimate the modular inverse of k bits requires $30 * (2 * (448)^2 + 5 * 448 + 1) = 12,109,470$ single-precision multiplications. Thus total overhead is $33,892,172$ single-precision multiplications. We conclude that the overhead of the PkQ decryption is 14.8 modular multiplications of 1024 bits with the pre-computation of $p^{-1} \bmod q$ and p^2. The details of the estimation is as follows:

Theorem 3 *From step 3 to step 9 of the PkQ decryption (algorithm 4), we need (1)*$34k^2 + 72k + 16$ *single-precision multiplications, and (2)2 modular inverses of* k

bits, where k is the bit-length of the prime p of q. If the parameters $(p^2)^{-1} \bmod q$ *and* p^2 *are not pre-computed, we need (1)*$35k^2+74k+17$ *single-precision multiplications, and (3)3 modular inverses of k bits.*

Proof: In step 3, we compute $F = (\pm M_p)^2 \bmod p^2$ and we assign them as $F_0 = (M_p)^2 \bmod p^2$ and $F_1 = (p-M_p)^2 \bmod p^2$. Because M_p and $p-M_p$ are smaller than p, the two values $(M_p)^2$ and $(p-M_p)^2$ are smaller than p^3. We need 2 multiplications of k bits, which is $2k^2+4k+2$ single-precision multiplications.

In step 4, we compute $E = C-F \bmod p^2$ and we assign them as $E_0 = C-F_0 \bmod p^2$ and $E_1 = C-F_1 \bmod p^2$. We need 2 modular reduction of $3k$ bits by $2k$ bits, which is $4k^2+6k$ single-precision multiplications.

In step 5, we compute $B = E/p$ and we assign them as $B_0 = E_0/p$ and $B_1 = E_1/p$. We need 2 divisions of $2k$ bits by k bits, which is $2k^2+6k$ single-precision multiplications.

In step 6, we compute $K = (\pm M_p) * B * (2 * F)^{-1} \bmod p$ and we assign them as $K_0 = (M_p) * B_0 * (2 * F_0)^{-1} \bmod p$ and $K_1 = (-M_p) * B_1 * (2 * F_1)^{-1} \bmod p$. The values F_0 and F_1 cannot be computed in advance. We need 2 modular inverses of k bits, and 4 modular multiplications of k bits, which is $8k^2+20k+4$ single-precision multiplications.

In step 7, we compute $A = (\pm M_p) + p * K$ and we assign them as $A_0 = M_p + p * K_0$ and $A_1 = M_p + p * K_1$. We need 2 multiplications of k bits and k bits, which is $2k^2+4k+2$ single-precision multiplications.

In step 8, we compute $V = (\pm M_q - A) * (p2\_inv\_q) \bmod q$ and we assign them as $V_{0,0} = (M_q - A_0) * (p2\_inv\_q) \bmod q$, $V_{1,0} = (M_q - A_1) * (p2\_inv\_q) \bmod q$, $V_{0,1} = (-M_q - A_0) * (p2\_inv\_q) \bmod q$, and $V_{1,1} = (-M_q - A_1) * (p2\_inv\_q) \bmod q$. We need 4 modular multiplications of k bits, which is $8k^2+20k+4$ single-precision multiplications.

Finally in step 9, we compute $M = A + (p^2) * V$ and we assign them as $M_{0,0} = A_0+(p^2)*V_{0,0}$, $M_{1,0} = A_1+(p^2)*V_{1,0}$, $M_{0,1} = A_0+(p^2)*V_{0,1}$, and $M_{1,1} = A_1+(p^2)*V_{1,1}$. We need 4 multiplications of $2k$ bits and k bits, which is $8k^2+12k+4$ single-precision multiplications. ■

4.4 Conclusion

In Table 1, we summarize the comparison among the HIME decryption, the RSA decryption, the Rabin based decryption, and the PkQ decryption [Tak98]. The dominant computation time of the HIME or the PkQ decryption is the computation of both $C^{(p+1)/4} \bmod p$ and $C^{(q+1)/4} \bmod q$, which is estimated 257.3 for 1344-bit

modulus. The numbers in the table are the number of the modular multiplication of 1024 bits.

Table 1: Comparison of efficiency among several schemes

1024-bit RSA	1024-bit Rabin	1344-bit HIME	1344-bit PkQ
385	386	275	272

We conclude that although the 1344-bit HIME decryption is faster than the 1024-bit RSA/Rabin decryption, it is slower than the 1344-bit PkQ decryption.

Note that in theorem 2 and theorem 3 we proved the decryption algorithm of the HIME cryptosystem is still slower than that of the PkQ cryptosystem even if their pre-computations are not carried out.

The secret keys of the HIME are p, q without pre-computation or $p, q, p^{-1} \bmod q, pq$ with pre-computation. The secret keys of the PkQ are p, q without pre-computation or $p, q, (p^2)^{-1} \bmod q, p^2$ with pre-computation. Therefore, the total secret key sizes of the HIME cryptosystem and the PkQ cryptosystem are same for the same bit-length modulus.

References

[BR94] M. Bellare and P. Rogaway, "Optimal asymmetric encryption - How to encrypt with RSA," Advances in Cryptology - EUROCRPT'94, LNCS 950, pp.92-111, 1994.

[Bon01] D. Boneh, "Simplified OAEP for the RSA and Rabin Functions," Advances in Cryptology – CRYPTO 2001, LNCS 2139, pp.275-291, 2001.

[BDG99] D. Boneh, G. Durfee, and N. Howgrave-Graham, "Factoring $N = p^r q$ for large r," Advances in Cryptology – CRYPTO'99, LNCS 1666, pp.326-337, 1999.

[Bre00] R. Brent, "Recent progress and prospects for integer factorisation algorithms," 6th Annual International Conference, COCOON 2000, LNCS 1858, pp.3-22, 2000.

[CUS02] K. CHIDA, S. UCHIYAMA, and T. SAITO, "A new factoring method of integers $N = p^r \times q$ for large r," IEICE Trans. Fundamentals, Vol.E85-A No.5 pp.1050-1053, 2002.

[ET02] P. Ebinger and E. Teske, "Factoring $N = pq^2$ with the Elliptic Curve Method," Technical Report of CACR, University of Waterloo, CORR 2002-02, 2002.

[ECM98] ECMNET Project, `http://www.loria.fr/~zimmerma/records/ecmnet.html`

[HIME01] HIME(R), Specification and Self-Evaluation Report, Hitachi Ltd. `http://www.sdl.hitachi.co.jp/crypto/hime/`

[MvOV97] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of applied cryptography*, CRC Press, 1997.

[LV01] A. Lenstra, E. Verheul, "Selecting cryptographic key sizes," Journal of Cryptology, 14 (4), p.255-293, 2001.

[OS01] K.Okeya and K.Sakurai, "Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve", *CHES2001*, LNCS 2162, pp.126-141, Springer-Verlag, 2001.

[PO96] R. Peralta and E. Okamoto, "Faster factoring of integers of a special form," IEICE Trans. Fundamentals, Vol.E79-A, No.4, pp.489-493, 1996.

[PKCS] Public-Key Cryptography Standards, PKCS # 1, RSA Laboratories, `http://www.rsasecurity.com/rsalabs/pkcs/`

[RSA155] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. Leyland, J. Marchand, F. Morian, A. Muffett, C. Putnam, C. Putnam, and P. Zimmermann, "Factorization of a 512-Bit RSA Modulus," Advances in Cryptology – EUROCRYPT 2000, LNCS1807, pp.1-18, 2000.

[Sho01a] V. Shoup, "OAEP reconsidered," Advances in Cryptology – CRYPTO 2001, LNCS 2139, pp.239-259, 2001.

[Sho01b] V. Shoup, "A proposal for an ISO standard for public key encryption," `http://shoup.net/`

[Sil00] R. Silverman, "A cost-based security analysis of symmetric and asymmetric key lengths," RSA Laboratories Bulletin, No.13, (2000). `http://www.rsasecurity.com/rsalabs/bulletins/bulletin13.html`

[Tak98] T. Takagi, "Fast RSA-Type Cryptosystem Modulo $p^k q$," Advances in Cryptology - CRYPTO '98, LNCS 1462, pp.318-326, 1998.

[Tak01] T. Takagi, "New Public-Key Cryptosystem with Fast Decryption," PhD Thesis, Technische Universtät Darmstadt, Germany, 2001. (available from `http://elib.tu-darmstadt.de/diss/000104/`)