Evaluation of Security Level of Cryptography

HIME-1

Dan Boneh **Mihir Bellare** **Phillip Rogaway**

January 16, 2001

Contents

This document analyzes HIME-1. It is based on [1] and [2].

1 Summary

Misclassification. HIME-1 is a public-key encryption scheme. The functionality it delivers is data confidentiality. It was correctly placed in the Asymmetric Cryptographic Schemes category, but misclassified with regard to Security functions of asymmetric cryptographic schemes, where key-sharing, rather than confidentiality, was marked. Although a public-key cryptosystem like HIME-1 is a tool that can be used in a key-sharing protocol, it does not by itself provide key-sharing functionality.

Scheme. Let $N = p^d q$ and let k be the binary length of pq, where p, q are primes congruent to $3 \bmod 4$ and $d > 1$ is an odd integer like 3. HIME-1 encrypts a message by first applying the OAEP transform to get a point x and then applying the trapdoor one-to-one function: $x \to [x^{2N} \bmod N, \left(\frac{x}{N}\right)]$ where $x \in \{0,1\}^{k-2}$.

Performance claims. The motivation behind the design of HIME-1 is to have very fast decryption, even if at the cost of slowing encryption. It is claimed that the decryption time of HIME-1 is less than that of any other cryptosystem of comparable security. The instantiation used to make this claim sets the length of p, q to each be 256 bits, and $d = 3$, so that the modulus is 1024 bits.

Security claims. It is claimed that HIME-1 is proven secure in the random-oracle model assuming the hardness of the problem of factoring integers of the form N above. The notion of security targeted is indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2).

Strengths. The proposal has the following strengths. It targets an important primitive which has a clear and well-defined security goal, namely asymmetric encryption achieving IND-CCA2. (CRYPTREC should eventually include some primitive meeting this goal. The question is whether HIME-1 should be it.) The proposal attempts to prove security, and on the whole is clearly and concisely written. (Some editorial comments on the manuscript can be found in Section 2.5 below.)

Critiques. The proposal has the following weaknesses.

1. The suggested parameters, meaning $|p| = |q| = 256$ and $d = 3$, do not appear to offer a margin of security sufficient for a long-term encryption standard.

2. It is unclear that HIME-1 offers advantages over its competitors in terms of performance. Encryption is substantially slower than in other schemes, and decryption does not appear to be faster than for elliptic curve schemes of comparable security, or the Rabin scheme using the same modulus as HIME-1.

3. There are alternatives with comparable decryption speed which do not pay a high price in encryption speed.

4. The given proof of security inherits the recently discovered gap in the proof of chosen-ciphertext security for OAEP [5]. The property can be resurrected by switching to the OAEP+ embedding method of that same paper, so this concern is ultimately less important than the ones above.

On balance, there would seem to be alternative systems that are superior to HIME-1.

2 Detailed Evaluation

We now provide a full evaluation of the performance and security of the proposed system.

2.1 Parameter choices

The instantiation of HIME-1 suggested in the proposal and used in the implementation has a 1024-bit modulus $N = p^3q$ with p and q being 256-bits long. In the light of recent factoring trends, these parameters do not seem to offer a margin of security sufficient for a long term encryption standard.

A year ago Lygeros and Mizony used the ECM factoring method to find a 180-bit prime factor within 13 days using a single 500Mhz Dec Alpha. The ECM method has asymptotic running time of $L_p[1/2, \sqrt{2}]$ where p is the smallest prime factor of N. Using the asymptotic formula we see that finding a 256-bit factor should take approximately 48000 times the work done by Lygeros and Mizony. This comes out to be approximately 20 times the effort it took to factor RSA-155 (a 512 bit integer) using NFS. These numbers are only an estimate based on the asymptotic formulas and could be off by as much as a factor of 10. So, suppose that finding a 256-bit factor using ECM is 200 times the effort of factoring RSA-155 using NFS. This safety margin is far smaller than the margin provided by comparable standards. For example, factoring a regular 1024-bit RSA modulus $N = pq$ would take $3.1 \cdot 10^6$ times the work of factoring RSA-155 (assuming no memory-space constraints). See Appendix A for detailed calculation of factoring running times.

At a minimum, a 1536-bit modulus should be used so that, with $d = 3$, the primes p and q are each 384 bits long. Factoring such numbers using ECM takes approximately 10^6 times the work of factoring RSA-155 using NFS, so that security becomes comparable to that offered by RSA with a 1024-bit modulus.

However increasing the sizes of the numbers in this manner adversely impacts performance.

2.2 Performance

When using a 1024-bit modulus with $d = 3$, decryption requires two exponentiations modulo a 256-bit number. Following the discussion above, we claim that the security of this 1024-bit HIME-1 system is comparable to the security of ECC over the field $\mathbb{F}_{2^{131}}$. However, ECC decryption in this field is faster than HIME-1 decryption. In other words if HIME-1 is compared to its competitors at the same security level, it does not have the best decryption speed.

To compare HIME-1 to ECC in the field $\mathbb{F}_{2^{163}}$ one has to use a 1536-bit HIME modulus. Consequently, decryption amounts to two exponentiations modulo a 384-bit number. Again, ECC-163 decryption is faster than 1536-bit HIME-1 decryption.

In addition, HIME-1 encryption is significantly slower than encryption in competing systems. Encryption with HIME-1 takes a full exponentiation, while encryption with either a Rabin or RSA system can be just a few multiplications, and encryption in ECC involves multiplications (the equivalent of exponentiations) of much smaller numbers.

Replacing the HIME-1 one-way function by the comparable $f(x) = x^2 \bmod N$ as discussed in Section 2.4 below avoids the high encryption time, while maintaining the claimed decryption time, but still does not seem to beat ECC.

2.3 The proof of security

The HIME-1 system uses OAEP [3] to provide chosen ciphertext security in the random-oracle model. Recently Shoup shows that OAEP cannot apply to all trapdoor permutations [5]. For the RSA function it was later shown by Fujisaki et al. [4] that OAEP is valid. Unfortunately, Fujisaki et al.'s proof does not apply to the HIME-1 system since in HIME-1 the padded message length is much less than the size of the modulus. For example, for $d = 3$ the padded message length is a quarter of the length of the modulus. Consequently, one cannot rely on OAEP to provide chosen-ciphertext security for HIME-1. Instead, one should use OAEP+ described by Shoup [5]. The main difference between OAEP and OAEP+ is that the block of 0^{k_1} in OAEP is replaced by $W(m, r)$ where W is a hash function outputting bit strings in $\{0, 1\}^{k_1}$. This change should be made but will not impact performance much, so eventually is less important to the evaluation than the performance issues.

2.4 Alternatives and enhancements

Squaring. Inverting the HIME-1 trapdoor permutation is as hard as factoring the modulus. However, HIME-1 seems to perform worse than other trapdoor functions with this property. For example, let $N = p^d q$ and define $f_{SQR}(x) = [x^2 \bmod N, \left(\frac{x}{N}\right)]$ for $x \in [0, N/4]$. A standard argument shows that inverting

f_{SQR} is as hard as factoring N. In addition, inverting f_{SQR} takes comparable time to inverting the HIME-1 function. Given $y = x^2 \bmod N$ compute the square root of y modulo N as follows: (1) first compute the square root of y modulo p and q to obtain x_p and x_q respectively (this is analogous to the HIME-1 decryption algorithm), (2) next, use Hensel lifting on x_p to compute the square root of y modulo p^d; this gives x_{p^d}, (3) finally, use the CRT to combine x_q and x_{p^d} to obtain the required square root of y modulo N. Step (2) takes little time since each Hensel lift only requires one modular inversion, but does not require any exponentiations. Hence, inverting f takes comparable time to inverting the HIME-1 function. The rest of the system remains unchanged. Note that encryption using f_{SQR} is much faster than HIME-1. Furthermore, the message length can be almost as long as the modulus (in HIME-1 the message length must be much shorter than the modulus). Consequently, a system using f_{SQR} appears to be superior to HIME-1 in every respect.

Jacobi symbol. A HIME-1 ciphertext includes both $y = x^{2n} \bmod N$ and $\left(\frac{x}{N}\right)$. Consequently, the encryptor must compute the Jacobi symbol of x during the encryption process. Appending $\left(\frac{x}{N}\right)$ to the ciphertext is done so that the decryptor could choose from among the two $2N$'th roots of y modulo N that lie in $[0, 2^k]$. We claim that it is unnecessary to compute the Jacobi symbol during encryption. This is due to that fact that HIME-1 uses OAEP. Consider the modified HIME-1 system where the ciphertext is $y = x^{2n} \bmod N$ (without the Jacobi symbol). Then y has at most two $2n$'th roots modulo pq in the range $[0, 2^k]$. With very high probability only one of these roots will satisfy the OAEP test (testing that 0^{k_1} appears in the pad). Hence, the decryptor can choose the correct root among the two candidate roots of y. By using the OAEP test to distinguish between the two roots of y neither the encryptor nor the decryptor need to compute the Jacobi symbol of the ciphertext.

Primality testing. Section 3.3.3 of the specification describes the primality test used to generate primes for the HIME-1 system. The system requires primes satisfying $p = q = 3 \bmod 4$. Testing primality of such primes is done using a much simpler primality test than the one described in Section 3.3.3. Given a prime candidate $p = 3 \bmod 4$ test if it is prime as follows: (1) pick a random $g \in \mathbb{Z}_p^*$ and compute $x = g^{\frac{p-1}{2}} \bmod p$, (2) declare p to be prime if $x = \pm 1$, otherwise p is not prime. The primality test given in Section 3.3.3 reduces to this test for $p = 3 \bmod 4$. There is no reason to write the more complicated test in the proposal.

2.5 Editorial comments

Both the specification of HIME-1 and the self-evaluation report are written clearly. Below are a few comments that might help in improving the writeup.

Section 3.4 of the self-evaluation report includes a table of running times for various arithmetic operations. The row describing the computation of the Jacobi symbol is incomplete. Jacobi symbol times should be given for both when the factorization of N is known, and for when the factorization of N is unknown. Both cases are used by the HIME-1 system (one for encryption the other for decryption). Clearly computing the Jacobi symbol is much easier when the factorization of N is given. The current writeup does not say which of the two timing measurements is provided.

The relations given in Figure 1 are incomplete. One should show the separations between the entries in the last row and the other entries. They are quite easy to figure out.

Section 2.1 fails to provide a convincing argument that a scheme with improved decryption time is attractive even if encryption time goes up. Email decryption could be performed offline. And email decryption presents no significant performance issues anyway.

Section 2.2.3 lacks details on how decryption is performed in that it is not clear what are p^{-1} and q^{-1} and how arithmetic is being done in the exponents. This is clarified later in Section 3. A suggestion is that this double duplication is unnecessary. Instead, just move Section 3 up to Section 2 as the only description.

Section 2.2.4 should also discuss the ECM factoring method and its impact. There is only a brief discussion of this later in the self-evaluation report.

References

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

[2] Hitachi, Ltd. Self Evluation Report — HIME-1 CryptoSystem. Hitachi, Ltd., 2000.

[3] M. Bellare and P. Rogaway. Optimal asymmetric encryption — How to encrypt with RSA. Advances in Cryptology — Eurocrypt '94. Lecture Notes in Computer Science, vol. 950, 1994.

[4] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP is still alive! Record 00-061, IACR eprint library, `http://eprint.iacr.org`.

[5] V. Shoup. OAEP reconsidered. Record 00-060, IACR eprint library, `http://eprint.iacr.org`.

A Factoring Time Estimates

We present two calculations for showing that $N = p^3 q$ with both p and q being 256-bit long does not provide adequate security for a long-term standard.

Direct comparison of ECM(256) and NFS(1024) Let $N_0 = p^3 q$ be a 1024-bit modulus with p and q 256-bits each. Let $N_1 = p_1 q_1$ be a 1024-bit modulus with p_1 and q_1 512-bits each. We first compare the running time of the ECM method for factoring N_0 to the running time of NFS for factoring

N_1. Our conclusion is that (ignoring leading constants) factoring N_0 is easier than factoring N_1 by a factor of $1.5 * 10^6$. Since N_1 is the typical value used in standards, the conclusion is that N_0 provides insufficient security.

Let $T_{ECM}(n)$ and $T_{NFS}(n)$ denote the times taken by the ECM and NFS methods, respectively, to factor $n = p^d q$. Then

$$\begin{aligned}
\ln(T_{ECM}(n)) &= \ln(C_{ECM}) + \sqrt{2} \cdot (\ln(p))^{1/2} \cdot (\ln\ln(p))^{1/2} \\
\ln(T_{NFS}(n)) &= \ln(C_{NFS}) + 1.9 \cdot (\ln(n))^{1/3} \cdot (\ln\ln(n))^{2/3} \; .
\end{aligned}$$

Here C_{ECM} and C_{NFS} are, respectively, the constants in the big-oh factors in the two algorithms. We want to compare these two times. Let $s = d + 1$. Then $|n| = s \cdot |p|$. We want to know the ratio of the running times. With this in mind we take the logarithm of the ratio of the running times:

$$\begin{aligned}
&\ln \frac{T_{ECM}(n)}{T_{NFS}(n)} \\
&\quad = \ln(T_{ECM}(n)) - \ln(T_{NFS}(n)) \\
&\quad = \ln(C_{ECM}) + \sqrt{2} \cdot (\ln(p))^{1/2} \cdot (\ln\ln(p))^{1/2} - \ln(C_{NFS}) - \\
&\qquad\qquad 1.9 \cdot (\ln(n))^{1/3} \cdot (\ln\ln(n))^{2/3} \\
&\quad = \ln \frac{C_{ECM}}{C_{NFS}} + \sqrt{2} \cdot (\ln(p))^{1/2} \cdot (\ln\ln(p))^{1/2} - 1.9 \cdot (\ln(n))^{1/3} \cdot (\ln\ln(n))^{2/3} \\
&\quad \approx \ln \frac{C_{ECM}}{C_{NFS}} + \sqrt{2} \cdot (\ln(2^{k/2}))^{1/2} \cdot (\ln\ln(2^{k/2}))^{1/2} - \\
&\qquad\qquad 1.9 \cdot (\ln(2^{sk/2}))^{1/3} \cdot (\ln\ln(2^{sk/2}))^{2/3} \\
&\quad = \ln \frac{C_{ECM}}{C_{NFS}} + \sqrt{k\ln(2)} \cdot (\ln\ln(2^{k/2}))^{1/2} - 1.9 \cdot \left(\frac{sk\ln(2)}{2}\right)^{1/3} \cdot (\ln\ln(2^{sk/2}))^{2/3} \; .
\end{aligned}$$

Now we plug in $k = 512$ and $s = 4$, meaning $d = 3$ and $|p| = |q| = 256$. We get

$$\begin{aligned}
&\ln \frac{T_{ECM}(n)}{T_{NFS}(n)} \\
&\quad \approx \ln \frac{C_{ECM}}{C_{NFS}} + \sqrt{512\ln(2)} \cdot (\ln\ln(2^{256}))^{1/2} - 1.9 \cdot (1024\ln(2))^{1/3} \cdot (\ln\ln(2^{1024}))^{2/3} \\
&\quad \approx \ln \frac{C_{ECM}}{C_{NFS}} - 16.55 \; .
\end{aligned}$$

For simplicity let's assume that the ratio of the constants is one. In that case we are saying that the ECM method will be $e^{16.5} \approx 1.5 * 10^6$, ie. 1.5 million, times faster than the NFS. In other words, factoring numbers of the form used by HIME-1 with the choices $k = 512$ and $d = 3$ is less secure than 1024-bit RSA by about this factor.

Comparison based on the time to factor RSA-155 Our second estimate is based on current records for factoring using ECM and NFS. A year ago Lygeros and Mizony used the ECM factoring method to find a 180-bit prime factor within 13 days using a single 500MhZ Dec Alpha. We use the asymptotic running time of ECM to estimate the time it takes to find a 256-bit prime factor. The direct ratio of running times is:

$$\frac{L_{1/2,\sqrt{2}}(2^{256})}{L_{1/2,\sqrt{2}}(2^{180})} = 3371$$

However, one must also take into account that arithmetic modulo a 1024 bit number takes longer than arithmetic modulo a 421-bit number (the size of the modulus used by Lygeros and Mizony). Assuming all arithmetic operations takes cubic time, we need to add a factor of $(1024/421)^3 = 14.4$ to the estimate. This implies that finding a 256-bit factor of a 1024-bit modulus should take approximately 48000 times the work of finding a 180-bit factor of a 421-bit modulus. Hence, factoring $N_0 = p^3 q$ should take approximately 1727 years on a single 500MhZ Dec Alpha.

We compare the effort of factoring N_0 using ECM to the effort of factoring RSA-155 which took 3.7 months using 300 machines. We see that factoring N_0 using ECM takes about 20 times the work of factoring RSA-155 using NFS. These numbers are only an estimate based on the asymptotic formulas and could be off by as much as a factor of 10. So, suppose that finding a 256-bit factor using ECM is 200 times the effort of factoring RSA-155 using NFS. This safety margin is far smaller than the margin provided by comparable standards. For example, factoring a regular 1024-bit RSA modulus $N = pq$ would take $3.1 \cdot 10^6$ times the work of factoring RSA-155 (assuming no memory-space constraints).