HIME-2 の安全性に関する詳細評価報告

2001 年 1 月

Ver 3.0 2000/12/27(Wed)

HIME-2 とは, CRYPTREC によって 2000 年 6 月 13 日にアナウンスされた暗号公募に対して, (株) 日立製作所が応募した守秘用途公開鍵暗号アルゴリズムである. 本稿は応募暗号に対する詳細評価の一環として, 提案者から提出された暗号技術仕様書 (以下, 仕様書) および自己評価書 (以下, 評価書) の記述に基づいて, HIME-2 の安全性を議論することを目的としている.

安全性を議論するためには暗号アルゴリズムが明確に特定されていなければならない. 記述の明確さについては議論の余地があるが, 本稿では明確さの基準として, 仕様書とそこで指示された公開情報からプログラムを作成した 2 つのパーティ間で暗号通信が実現できるレベルの記述をもって明確であると呼ぶことにしたい. この意味では, HIME-2 は仕様書およびそこで指示された公開情報からでは暗号アルゴリズムを一意的に特定できない という問題点を有している. したがって HIME-2 の安全性評価は不可能と結論づけることも可能である.

しかし HIME-2 が既にスクリーニング評価を通過し, 詳細評価の対象になっている現状を考慮すれば[1], 何らかの形で安全性評価を行う必要があると考える. そこで本稿では, 安全性評価を行うための修正アルゴリズムを提示することとした. ここで提示する修正アルゴリズムはあくまで評価者の推測であって, 提案者の意図するアルゴリズムではない可能性が高い. しかし修正アルゴリズムは, 詳細は記述しないものの, 仕様書を善意的に解釈した上で, 暗号的見地から妥当と思われる形で修正を行った. 繰り返しになるが, このような修正は望ましくないのは明らかであるが, 仕様書に記載されたアルゴリズムが一意的に特定でない一方で, HIME-2 がスクリーニング評価を通過し, 詳細評価として安全性評価が必要であるということを考えれば, このような修正を施さざるを得ない.

本稿の構成は以下の通りである. まず第 1 節では, HIME-2 の仕様書にて定義されるアルゴリズムについて検討する. 仕様書の記述には暗号アルゴリズムが一意的に特定できないといった問題点があるため, 問題点を指摘した後に, 安全性評価を行うための修正アルゴリズムを提示する. 修正アルゴリズムは評価者の推測であって, 提案者の意図するアルゴリズムではないことに注意されたい.

第 2 節では HIME-2 の修正アルゴリズムに基づいて, 暗号スキームと暗号プリミティブの両面から安全性評価を行う. 暗号スキームに関しては HIME-2 は証明可能安全性であり, IND-CCA2 を達成していると評価書で主張されているが, 2 つの理由により, そのレベルの安全性は達成できていないと考える. すなわち, 評価書の証明では HIME-2 が IND-CCA2 の安全性を持つことを説明できない ことを示す. また暗号プリミティブのレベルでは, 現在での安全性には問題ないものの, 将来的な安全性については, 鍵長などに可変性がないため, 近い将来には解読可能になり得ることを指摘する. さらに補助的に使用される関数の安全性も高くはないことを指摘する.

第 3 節では HIME-2 で設定されたパラメータ値の妥当性について議論する. 仕様書ではさまざまなパラメータの具体的な値が示されているが, 設定理由が明確でない場合が多いため, あえてこのような議論を行った. 結論として妥当と思われる根拠を推測することができる一方で, いくつかのパラメータについては根拠が不明であり, さらなる議論が必要であることを指摘する.

以下では, 仕様書および評価書に記載された文言は枠線で囲んで引用することにする.

[1] スクリーニング評価の具体的な判断基準は公開されていないが, 暗号アルゴリズムが一意に特定できないアルゴリズムが通過できるという意味では, スクリーニング評価結果の信憑性に疑問を呈せざるを得ない.

1 アルゴリズム仕様

HIME-2 の安全性評価において最も問題となるのは, 提案者によって提出された暗号技術仕様書に記載された情報のみでは, 暗号アルゴリズムが一意的に特定できない点である. この理由として,

1. 仕様詳細が記述されていない.
2. 仕様詳細が記述されているが, 意味をなしていない.
3. 仕様の表記に不整合が見受けられる.
4. 記述されたアルゴリズム等に (数学的に) 誤った記述が見受けられる.
5. 記述意図が不明.
6. 誤植

などが挙げられる. 以下 1.2 節から 1.6 節にかけて項目別にコメントする. しかし安全性評価を実施するためには, 暗号アルゴリズムが特定できなくてはならない. そこで本稿では 1.7 節において修正アルゴリズムを提示することとする. 修正内容の詳細は記述しないが, 仕様書を善意的に解釈した上で, 暗号的見地から妥当と思われる形で修正を行った.

なお評価者は安全性評価を目的としているため, 自己評価書第 2 節以降の内容に関しては吟味しておらず, すなわちその内容の正当性は判断していないことを注意しておく.

1.1 仕様詳細が記述されていない

1.1.1 関数 G_2, H_2 (仕様書 3.2.6 節)

仕様書 3.2.6 節 The functions G_2, H_2
C : some constant (512-bit)
$C_1, \cdots, C_7$: some constants (128-bit)
h : a hash function $\{0,1\}^{\infty} \to \{0,1\}^{160}$ (SHA-1)
$h'(x) := h((x||x) \oplus C)$ の上位 128-bit : $\{0,1\}^{256} \to \{0,1\}^{128}$
$G_2 : \{0,1\}^{128} \to \{0,1\}^{896}$, $G_2 := h'(x||C_1)||h'(x||C_2)||\cdots||h'(x||C_7)$
$H_2 : \{0,1\}^{896} \to \{0,1\}^{128}$, $H_2 := h'(x_1||C_1) \oplus h'(x_2||C_2) \oplus \cdots \oplus h'(x_7||C_7)$ (ただし $x = x_1||x_2||\cdots||x_7$)

仕様書 3.2.6 節では, 関数 G_2, H_2 の計算アルゴリズムが示されている. アルゴリズム詳細中では定数 $C, C_1, \ldots, C_7$ の使用が規定されているが, 仕様書ではその具体的な値が示されていない. この関数は一般的な関数とはいえず, $C, C_1, \ldots, C_7$ が規定されない限り, 関数 G_2, H_2 の計算をはじめとして, 暗号化・復号化が一意的に行えないという問題が生じる. そればかりでなく, 例えばこれら全ての定数を同じ値に設定した場合, 明らかに関数 G_2 は出力に偏りが生じてしまうなど, 安全性に対しても影響を与えることが予想される.

上記アルゴリズム詳細の表記に対するコメントを以下に示す: まず, ハッシュ関数 h の定義域が $\{0,1\}^{\infty}$ となっているが, これでは無限の長さのビット列を意味してしまい, SHA-1 の仕様とは異なってしまう. SHA-1 の仕様を考慮すれば, ここでは任意長のビット列を意味する $\{0,1\}^*$ にすべきである. 次に, 関数 H_2 での変数 $x_1, \ldots, x_7$ のビット長が指定されていない. $|x_i| = 128$ $(1 \le i \le 7)$ などの表記が必要である. さらに, 変数 x が, 128 ビット変数と 896 ビット変数の双方に使用されているのは読みにくい. 後者では別の変数 y などを用いるべきである.

1.1.2 convert(仕様書 3.2.4 節)

仕様書 3.2.4 節 convert
Input 平文 m, 乱数 R (192-bit)
Output m' (1024-bit)

1. $r =$ most significant 128-bit of $h(R)$.
2. $M = (m||0^{128}) \oplus G_2(r)$
3. $m' = M||(r \oplus H_2(M))$
4. Return m' and end.

仕様書 3.2.4 節では関数 convert の計算アルゴリズムが示されている. m のビット長が記述されていないが, 3.2.2 節の記述 $0 < m < 2^{768}$ をもとにすれば, ビット列と数字の 2 進数表記が同一視されていると解釈した上で, 関数 convert の入力値 m も $0 < m < 2^{768}$ であると類推できる. しかし $0 < m < 2^{767}$ の場合, $m0^{k_1}$ と $G(r)$ のビット長が異なってしまう. おそらく何らかの形で 0 パディングするのであろうが, それに関する言及が見られないため, 暗号化・復号化が一意的に行えないという問題が生じる.

1.2 仕様詳細が記述されているが, 意味をなしていない

1.2.1 復号化アルゴリズム (仕様書 2.2.3 節)

仕様書 2.2.3 節 復号化
暗号文 C に対して,

$$x_i = C^{\frac{(p_i+1)}{4}} \bmod p_i \quad \text{for } 1 \le i \le d,$$

を計算し, 2^d 個の $\{\varphi(e_1x_1, e_2x_2, \ldots, e_dx_i)|e_i \in \{-1, 1\}\}$ に対して, $x'_i = s_i||t_i$ $(s_i \in \{0,1\}^{k-k_0}, t_i \in \{0,1\}^{k_0}, 1 \le i \le 2^d)$ とするとき,

$$m = \begin{cases} [x']^{k-k_0-k_1} & \text{if } [G(H(s_i) \oplus t_i) \oplus s_i]_{k_1} = 0^{k_1}, \\ \text{"reject"} & \text{otherwise.} \end{cases}$$

により, 平文 m の復号化を行う. 但し φ は, Chinese remainder theorem による環同型写像

$$\varphi : \mathbf{Z}_p \oplus \mathbf{Z}_q \xrightarrow{\sim} \mathbf{Z}_{pq}$$

を表す. また, $[a]^k, [a]_k$ は各々 a の上位および下位 k ビットを表す.

仕様書 2.2.3 節では HIME-2 の復号化アルゴリズムが示されているが, 記述内容に不明な点が多い.

まず Chinese Remainder Theorem による環同型写像 φ の入力値の個数に関する記述が論理的でない. 上記アルゴリズムの 3 行目では関数 φ は $d(=4)$ 個の入力値を持つように記述されている[2] のに対し, 7 行目は 2 個の入力値を持つように記述されており, 首尾一貫性がない. 仕様書の他の部分から類推すれば, 7 行目を

$$\varphi : \mathbf{Z}_{p_1} \oplus \cdots \oplus \mathbf{Z}_{p_d} \xrightarrow{\sim} \mathbf{Z}_{p_1 \ldots p_d}$$

と修正すべきように思えるが, これはあくまで評価者の類推でしかない.

[2] $\varphi(e_1x_1, e_2x_2, \ldots, e_dx_i)$ とあるのは $\varphi(e_1x_1, e_2x_2, \ldots, e_dx_d)$ の誤植と解釈すべきであろう.

次に, 復号化のアルゴリズムの手順が不明瞭である. アルゴリズム 3 行目で $x'_i = ...$ を用いて m の値を指定している[3] が, i のループと "reject" の関係が判然としていない. つまり, if 文で otherwise の場合, "reject" はループの継続条件なのか復号結果なのかが判断できない. この問題は, アルゴリズム詳細の記述内容から判断できるはずであるが, アルゴリズム詳細の記述内容には本稿 1.2.2 節で指摘するような問題があるため, その判断が下しにくい.

さらには, if 文を満足するような x'_i が複数存在した場合の記述も見られない.

上記アルゴリズムの表記に対するコメントとして, e_i と x'_i で同じ添字 i を使用しているため, 意味がわかりにくい問題を指摘する. 仕様書の他の部分では i は $1 \leq i \leq d$ のように使用されているので, ここでは x'_i ではなく, x'_j などに置き換えるべきである.

1.2.2 復号化アルゴリズム詳細 (仕様書 3.2.3 節)

仕様書 3.2.3 節 復号化
Input 暗号文 C, 秘密鍵 $(p_1, p_2, p_3, p_4, z_1, z_2, z_3)$
Output 平文 m

1. For $i = 1, 2, 3, 4$, calculate:
1.1 $e = (p_i + 1)/4$.
1.2 $C_i = C \mod p_i$.
1.3 $m_i = (C_i)^e \mod p_i$
2. $M_1 = \varphi(m_1, m_2, p_1, p_2, z_1)$
3. $M_2 = \varphi(m_3, m_4, p_3, p_4, z_2)$
4. $M = \varphi(M_1, M_2, (p_1p_2), (p_3p_4), z_3)$
5. $(m, w) \leftarrow \text{convert}^{-1}(M)$, If $w = 0$, goto 18.
6. $(m, w) \leftarrow \text{convert}^{-1}(n - M)$, If $w = 0$, goto 18.
7. $M = \varphi(M1, (p_3p_4) - M_2, (p_1p_2), (p_3p_4), z_3)$
8. $(m, w) \leftarrow \text{convert}^{-1}(M)$, If $w = 0$, goto 18.
9. $(m, w) \leftarrow \text{convert}^{-1}(n - M)$, If $w = 0$, goto 18.
10. $M_2 = \varphi(p_3 - m_3, m_4, p_3, p_4, z_2)$
11. $M = \varphi(M_1, M_2, (p_1p_2), (p_3p_4), z_3)$
12. $(m, w) \leftarrow \text{convert}^{-1}(M)$, If $w = 0$, goto 18.
13. $(m, w) \leftarrow \text{convert}^{-1}(n - M)$, If $w = 0$, goto 18.
14. $M = \varphi(M1, (p_3p_4) - M_2, (p_1p_2), (p_3p_4), z_3)$
15. $(m, w) \leftarrow \text{convert}^{-1}(M)$, If $w = 0$, goto 18.
16. $(m, w) \leftarrow \text{convert}^{-1}(n - M)$, If $w = 0$, goto 18.
17. $M_1 = \varphi(p_1 - m_1, m_2, p_1, p_2, z_1)$, goto 3.
18. Return m and end.

仕様書 3.2.3 節では HIME-2 の復号化アルゴリズム詳細が示されている. このアルゴリズム詳細の大きな問題として, 仕様書 2.2.3 節の出力値として想定されている "reject" が出力されないようになっている点が挙げられる. つまり復号アルゴリズム (仕様書 2.2.3 節) と復号アルゴリズム詳細 (仕様書 3.2.3 節) で内容に一貫性が見られない. このことからアルゴリズム詳細には別の問題が生じる. 暗号文 C として, 例えば 1024 ビットのランダムな値

[3] $[x']^{k-k_0-k_1}$ とあるのは $[x'_i]^{k-k_0-k_1}$ の誤植と解釈すべきである.

を与えられた場合, C に対応する平文 m が存在しないときには上記アルゴリズムは停止せず, 永久ループに入ってしまうことがあり得る. この永久ループは提案者の意図であるかどうかは不明であるが, 意図していたならば復号アルゴリズムとしては不適であるし, 意図していないならば記述に大きな問題があると指摘せざるを得ない. しかも本稿 1.2.1 節で触れた通り, 復号アルゴリズムの記述にも問題があるため, 結果として復号アルゴリズムの意図が判断しにくい点も問題である.

上記アルゴリズム詳細の表記において, 数としての 0 と, 128 個の 0 が並んだビット表現としての $(0 \ldots 0)$ を同一視していることに触れるべきである.

1.3 仕様の表記に不整合が見受けられる

1.3.1 記号の説明 (仕様書 3.1 節)

仕様書 3.1 節 記号の説明

$x||y$: bit 列 x と bit 列 y の連結 (例:$(0110)||(101) = (0110101)$)
$J(a,b)$: Jacobi 記号 $\left(\frac{a}{b}\right)$. 3.3.5 参照.
$\varphi(x,y,m,n,z)$: $(x \bmod m, y \bmod n)$ に対し、$w \equiv x \bmod n,\ w \equiv y \bmod m$ なる $w \bmod mn$.
ただし、 $\gcd(m,n) = 1, z = n^{-1} \bmod m$. m,n,z を省略して書くこともある. 3.3.6 参照.
$f : \{0,1\}^a \to \{0,1\}^b$: f は入力が a-bit, 出力が b-bit の関数.
hash 関数 h は SHA-1 を用いる. 3.3.2 参照.

仕様書 3.1 節ではアルゴリズム詳細で使用される記号を定義している. 仕様書 3.3.6 節の記述内容から考えれば, 関数 $\varphi(x,y,m,n,z)$ の出力値 w の満たすべき条件は $w \equiv x \bmod n,\ w \equiv y \bmod m$ ではなく, $w \equiv x \bmod m,\ w \equiv y \bmod n$ の誤りであろう.

表記上の問題点として, 関数 φ の表記が非常に紛らわしい点が挙げられる. この関数が Chinese remainder theorem による準同型写像を意味する点は問題ないが, 仕様書 2.2.3 節では関数 φ を d 個 (つまり 4 個) の入力値を持つ関数として表記しているのに対し, 仕様書 3.1 節では 5 個の入力値を持つように表記している. しかも単に 5 個目の入力値を追加したわけでなく, 3 個目と 4 個目の入力値の意味を変更して使用している. 以上のことから仕様書 3.1 節での関数表記は, 仕様書 2.2.3 節の意図を考慮して $\varphi(x_1, \ldots, x_d)$ あるいは $\varphi(x_1, x_2, x_3, x_4)$ (この場合関数内容の説明も変更する必要が生じる), または仕様書 3.3.6 節の記述を考慮して $\mathrm{PHI}(x,y,m,n,z)$ とすべきである.

1.3.2 アルゴリズム詳細 (仕様書 3.2 節)

仕様書 3.2 節 HIME-2 公開鍵暗号アルゴリズム詳細

本章では, HIME-2 公開鍵暗号アルゴ リズムの実現方法について説明する. なお, 実際の使用を想定して, 鍵長を 1024 ビット ($|n| = 1024$) とし, 生成する素数 $p_i (1 \leq i \leq d)$ の個数を $d = 4$ としている.
また, HIME-2 復号化処理における

$$x_i = C^{\frac{(p_i+1)}{4}} \bmod p_i, \quad \text{for } 1 \leq i \leq d,$$
$$\varphi(x_p, x_q), \varphi(-x_p, x_q), \varphi(x_p, -x_q), \varphi(-x_p, -x_q)$$

の計算の効率化を考え, 鍵生成時に,

$$z_1 = p_2^{-1} \bmod p_1, \quad z_2 = p_4^{-1} \bmod p_3, \quad z_3 = (p_3 p_4)^{-1} \bmod (p_1 p_2)$$

を予め計算しておき, 秘密鍵の一部としている.

本稿 1.3.1 節でも指摘した通り, 関数 φ の入力値は d 個である. したがって上記 5 行目は

$$\varphi(\pm x_1, \ldots, \pm x_d)$$

または, 具体的に

$$\begin{array}{llll}
\varphi(x_1, x_2, x_3, x_4), & \varphi(-x_1, x_2, x_3, x_4), & \varphi(x_1, -x_2, x_3, x_4), & \varphi(-x_1, -x_2, x_3, x_4), \\
\varphi(x_1, x_2, -x_3, x_4), & \varphi(-x_1, x_2, -x_3, x_4), & \varphi(x_1, -x_2, -x_3, x_4), & \varphi(-x_1, -x_2, -x_3, x_4), \\
\varphi(x_1, x_2, x_3, -x_4), & \varphi(-x_1, x_2, x_3, -x_4), & \varphi(x_1, -x_2, x_3, -x_4), & \varphi(-x_1, -x_2, x_3, -x_4), \\
\varphi(x_1, x_2, -x_3, -x_4), & \varphi(-x_1, x_2, -x_3, -x_4), & \varphi(x_1, -x_2, -x_3, -x_4), & \varphi(-x_1, -x_2, -x_3, -x_4)
\end{array}$$

とすべきである.

1.3.3 鍵生成 (仕様書 3.2.1 節)

仕様書 3.2.1 節 鍵生成 ($|n| = 1024$)
Input 生成する鍵長 (n のビット長)
Output **公開鍵** n, **秘密鍵** $(p_1, p_2, p_3, p_4, z_1, z_2, z_3)$.

1. Choose random prime numbers $p_i (\equiv 3 \bmod 4), |p_i| = 256 (i = 1, 2, 3, 4)$.
2. $n = p_1 p_2 p_3 p_4$.
3. $z_1 = p_2^{-1} \bmod p_1$.
4. $z_2 = p_4^{-1} \bmod p_3$.
5. $z_3 = (p_3 p_4)^{-1} \bmod (p_1 p_2)$.
6. Return $(n, (p_1, p_2, p_3, p_4, z_1, z_1, z_2, z_3))$ and end.

生成する素数 p_i $(i = 1, 2, 3, 4)$ は, $n = p_1 p_2 p_3 p_4$ の素因数分解が困難となるように選ぶ必要があり, 以下のような条件を満足することが推奨されている.

- $p_i - 1$ が大きな素数 r_i を含む.
- $p_i + 1$ が大きな素数 s_i を含む.
- $r_i - 1$ が大きな素数 t_i を含む.

これらの条件を満たす素数 p, q の生成方法に関しては, 文献 [17] 等に示されており, そちらを参照されたい.

仕様書 3.2.1 節では鍵生成アルゴリズム詳細が示されている. しかし上記アルゴリズムで n の鍵長を入力値としておきながら, アルゴリズム内でその情報は使用されておらず, $|p_i| = 256$ としているのはアルゴリズムとして不適切である. [4] ここでは入力値を NULL とすべきである.

上記アルゴリズム詳細の表記に対し, 以下のコメントを行う: まず, 「素数 p, q」の表記は「素数 p_1, p_2, p_3, p_4 」の誤記と解釈すべきである. 次に, 手順 1. で "random prime" という表記があるが, それ以降で $p_i \equiv 3 \pmod 4$ という条件を付加しているため, random とは言えない. また素数 p_i の条件の中に s_i, t_i が登場するが, この記号は復号アルゴリズムで別の意味に使用されており, 混乱はきたさないものの, 使用を控えた方が好ましい.

1.3.4 暗号化 (仕様書 3.2.2 節)

仕様書 3.2.2 節 暗号化
Input 平文 m $(0 < m < 2^{768})$, 公開鍵 n
Output 暗号文 C

1. $m' = \mathrm{convert}(m, (R), 128)$
2. $C = m'^2 \bmod n$.
3. Return C and end.

仕様書 3.2.2 節では暗号化アルゴリズム詳細が示されている. しかし関数 convert への入力値 (R) が何を表しているのかが記述されていない点で不備がある (ただし関数 convert の仕様から乱数を意味するであろうことは類推可能である).

[4] 鍵生成アルゴリズムで, あたかも鍵長が設定できるような関数仕様でありながら, 実際は鍵長が固定である点が問題である.

またの関数 convert の仕様 (仕様書 3.2.4 節) では 2 個の入力値が想定されているのに対し, 上記アルゴリズムでは 3 個の入力値が与えれており, 不整合が見受けられる. 関数 convert 内で 128 というパラメータが使用されていると解釈できることから, ここでは $m' = \mathrm{convert}(m, (R))$ などとすべきであろう.

1.4 記述されたアルゴリズム等に (数学的に) 誤った記述が見受けられる

1.4.1 素数生成 (仕様書 3.3.3 節)

仕様書 3.3.3 節 素数生成
MILLER-RABIN(n,t)
Input $n \geq 3$ なる奇整数, $t \geq 1$.
Output 判定結果 (”prime” or ”composite”)
(アルゴリズム詳細は省略)

仕様書 3.3.3 節では素数生成アルゴリズムとして, 関数 MILLER-RABIN が示されている. しかしこの記述内容に関しては 2 つの問題点がある. まず本節のアルゴリズムでは, 合成数または疑素数の判定しかできないため, 出力値 ”prime” は誤りであり, 正しくは ”pseudo-prime” などとすべきである. このとき入力値 t は擬素数が素数である確率を制御するパラメータと考えられるが, その確率に関する記述は不可欠である. 次に, 本節のアルゴリズムは合成数か擬素数かを判定する内容であり, 素数生成そのものの機能は有していない. したがって素数生成を行うには, (MILLER-RABIN 関数をコールするような) 素数生成関数の定義が必要である. 特に HIME-2 の鍵生成スキームでは素数生成関数が必要となるが, 素数に対する付随条件が課されているため, 詳細な素数生成関数のアルゴリズム表記は不可欠である.

1.4.2 Jacobi 記号 (仕様書 3.3.5 節)

仕様書 3.3.5 節 Jacobi 記号の計算方法
JACOBI(a,n)
Input an odd integer $n \geq 3$, and an integer $a, 0 \leq a < n$
Output the Jacobi symbol

1. If $a = 0$ then return(0).
2. If $a = 1$ then return(1).
3. If a is even then $j \leftarrow \mathrm{JACOBI}(a/2, n) \cdot (-1)^{n^2-1/8}$;
   otherwise, $j \leftarrow \mathrm{JACOBI}(n \bmod a, a) \cdot (-1)^{(a-1)(n-1)/4}$.
4. return(j).

仕様書 3.3.5 節では Jacobi 記号の計算アルゴリズムが示されているが, 記述に誤りが見受けられる. 3. で a が偶数の場合に j に代入する値は, $\mathrm{JACOBI}(a/2, n) \cdot (-1)^{n^2-1/8}$ ではなく, $\mathrm{JACOBI}(a/2, n) \cdot (-1)^{(n^2-1)/8}$ である.

なお関数 JACOBI は, 仕様書 2.2.4 節でコメントされてる HIME-2 の変型アルゴリズムで使用されることが予想できるが, そうでない場合には必ずしも必要ない関数である.

1.4.3 Chinese Remainder Theorem による環同型写像 (仕様書 3.3.6 節)

仕様書 3.3.6 節 Chainese remainder theorem による環同型写像 φ の計算方法
PHI(x, y, n, m, z)
Input positive integers x and y, positive integers m and n (gcd(m,n)=1), a positive integer $z = m^{-1} \bmod n$,
Output w such that $w \in \mathbf{Z_{mn}}$ and $w \equiv x \bmod m, w \equiv y \bmod n$

1. $u \leftarrow x - y \bmod m$.

3. $v \leftarrow zu \bmod m$.

4. $h \leftarrow vn \bmod mn$.

5. return($y + h \bmod mn$).

仕様書 3.3.6 節では Chinise Remainder Theorem による環同型写像 φ の計算方法が示されているが, 記述に誤りが見受けられる. 関数の機能から考えて, 入力値で z の満たすべき条件は, $z = m^{-1} \bmod n$ ではなく, $z = n^{-1} \bmod m$ とすべきである.

また, アルゴリズム内の計算手順 2. が記載されていないが, この意図は解釈しかねる.

1.5 記述意図が不明

1.5.1 備考 (仕様書 2.2.4 節)

仕様書 2.2.4 節 備考
HIME-2 における公開鍵暗号 n の素因数分解困難性を考慮して, n のサイズは $|n| \geq 1024$ が望ましい. また, $|n| = 1024$ のとき, $d = 4 \sim 6$ 程度に設定する.
また, 上記復号化処理において, 正しい平文の探索の補助として,

$$w = \begin{cases} 0 & \text{if } 0 < x < n/2, \\ 1 & \text{if } n/2 \leq x < n \end{cases}$$

なる w, または, Jacobi 記号 $a = \left(\frac{x}{n}\right)$ を暗号文 C と共に送信することも可能である.

仕様書 2.2.4 節では備考として n のサイズ, d の大きさ, 補助情報 w の存在可能性について触れている.

HIME-2 の仕様として n, d の具体的な値が含まれるのならば, 本節で $|n| > 1024$, $d = 5, 6$ について触れる必要は小さい. また, HIME-2 の仕様として n, d の値は選択できるならば, その旨を明記した上で, アルゴリズムと n, d の具体的な値の依存性を切り分けるべきである.

また, Jacobi 記号による補助情報 w の送信可能性に触れているものの, w を利用したアルゴリズムは一切提示されておらず, HIME-2 の仕様に含まれるかどうかも判然としない.

いずれにしても仕様書 2.2.4 節の備考内容は, 仕様書・評価書のこれ以外の部分で記述が見受けられないため, 説明意図が不明であると言わざるを得ない.

1.6 誤植

前節までに指摘した誤植以外に, 以下の誤植が見受けられる.

1.6.1 最大公約数および逆元 (仕様書 3.3.4 節)

仕様書 3.3.4 節 最大公約数および逆元の計算方法
BINARY-EUCLID(a,n)
Input positive integers x and y
Output integers a, b and v such that $ax + yb = v$, where $v = gcd(x, y)$
(アルゴリズム詳細は省略)

仕様書 3.3.4 節では最大公約数および逆元の計算アルゴリズムが示されているが, 関数の宣言が内容と不整合である. 正しくは BINARY-EUCLID(x,y) であろう.

1.6.2 HIME-2 の安全性 (評価書 1.1 節)

評価書 1.1 節 HIME-2 の安全性 (抜粋)
HIME-2 が適応的選択暗号文攻撃に対して強秘匿 (IND-CCA2) であるとは, 任意の確率的多項式時間アルゴリズム Adv, M, 任意の定数 c, 十分大きな k, に対して,

$$\Pr(Adv^{D_{sk},G,H}(1^k, pk, m_0, m_1, C) = m | G, H \xleftarrow{R} \Omega; (pk, sk) \xleftarrow{R} \mathcal{G}_1(1^k);$$
$$(m_0, m_1) \xleftarrow{R} M^{D_{sk},G,H}; m \xleftarrow{R} \{m_0, m_1\}; C \xleftarrow{R} \mathcal{E}_1^{G,H}(pk, m)) < \frac{1}{2} + \frac{1}{k^c},$$

が成立することである. 但し Ω は $\{0,1\}^*$ から $\{0,1\}^\infty$ への写像全体を表す.

評価書 1.1 節の IND-CCA2 の定義の中で, $D_{s,k}$ とは何を示すのかが明記されていない.

1.7 修正アルゴリズム

前節までに指摘した通り, 提案者による HIME-2 の仕様書・評価書の記述には多くの問題点が見受けられる. 特に一意的に暗号化・復号化できない点は大きな問題であり, 安全性評価が不可能であるとも考えられる. しかしスクリーニング評価で大きな問題がないと判断されたという結果を尊重すれば, 詳細評価では何らかの安全性評価が必要である.

以上を踏まえ, 安全性評価を行う目的で, 評価者による修正アルゴリズムを提示する. ただしこの修正アルゴリズムはあくまでも評価者の推測であって, 提案者の意図するアルゴリズムではないことに注意されたい. また修正にあたっては, 暗号的見地から妥当な修正を施したつもりであるが, その妥当性の判断はここでは問題にしないことにする. このような修正は望ましくないが, 修正を施さない限り HIME-2 の安全性評価は不可能である.

以下, 修正方針を記す:

- 仕様書 3.2 節の記述を元に, 必要と思われる情報を加味する.
- 補助関数の具体的アルゴリズムは提示しない.
- 仕様書 2.2.4 節の変型アルゴリズムに関する内容は反映させない.

以下に修正アルゴリズムを提示する. なおアルゴリズム中では, 整数とその 2 進数表記は同一視して考えられており, 実装上では変換関数が必要となる.

1.7.1 鍵生成 (修正版)

Input 疑似乱数のシード $seed$
Output 公開鍵 n, 秘密鍵 $(p_1, p_2, p_3, p_4, z_1, z_2, z_3)$

1. For $i = 1, 2, 3, 4$, calculate:
   (a) generate a 256-bit prime number p_i with parameter $seed$
   (b) if $p_i \equiv 1 \pmod 4$ then goto 1.(a)
   (c) if p_i does not satisfy the follwing conditions, goto 1.(a)
   - $p_i - 1$ has a large prime factor r_i
   - $p_i + 1$ has a large prime factor
   - $r_i - 1$ has a large prime factor
2. $n = p_1 p_2 p_3 p_4$.
3. $z_1 = p_2^{-1} \bmod p_1$.
4. $z_2 = p_4^{-1} \bmod p_3$.
5. $z_3 = (p_3 p_4)^{-1} \bmod (p_1 p_2)$.
6. Return $(n, (p_1, p_2, p_3, p_4, z_1, z_1, z_2, z_3))$ and end.

注意 素数 p_i に課している 3 つ条件にはあいまいさが残るが, ここでは問題にしないことにする.

1.7.2 暗号化 (修正版)

Input 平文を表す 768-bit のビット列 m, 公開鍵 n (1024-bit), 疑似乱数の種 $seed_e$
Output 暗号文を表すビット列 C (1024-bit)

1. Generate a pseudo-random number R with parameter $seed_e$
2. $m' = \mathrm{convert}(m, R)$
3. $C = m'^2 \bmod n$.
4. Return C and end.

1.7.3 復号化 (修正版)

Input 暗号文を表すビット列 C (1024-bit), 秘密鍵 $(p_1, p_2, p_3, p_4, z_1, z_2, z_3)$
Output 平文を表すビット列 m (768-bit) または "reject"

1. For $i = 1, 2, 3, 4$, calculate:
   (a) $e = (p_i + 1)/4$.
   (b) $C_i = C \bmod p_i$.
   (c) $m_i = (C_i)^e \bmod p_i$.
2. $(m, w) \leftarrow \mathrm{decrypt}(m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
3. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, m_2, m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
4. $(m, w) \leftarrow \mathrm{decrypt}(m_1, -m_2, m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
5. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, -m_2, m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
6. $(m, w) \leftarrow \mathrm{decrypt}(m_1, m_2, -m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
7. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, m_2, -m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].

8. $(m, w) \leftarrow \mathrm{decrypt}(m_1, -m_2, -m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
9. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, -m_2, -m_3, m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
10. $(m, w) \leftarrow \mathrm{decrypt}(m_1, m_2, m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
11. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, m_2, m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
12. $(m, w) \leftarrow \mathrm{decrypt}(m_1, -m_2, m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
13. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, -m_2, m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
14. $(m, w) \leftarrow \mathrm{decrypt}(m_1, m_2, -m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
15. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, m_2, -m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
16. $(m, w) \leftarrow \mathrm{decrypt}(m_1, -m_2, -m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
17. $(m, w) \leftarrow \mathrm{decrypt}(-m_1, -m_2, -m_3, -m_4, p_1, p_2, p_3, p_4, z_1, z_2, z_3)$. If $w = 0$ then goto [end].
18. return("reject") and end.
19. [end] return m

1.7.4 convert(修正版)

Input ビット列 m (768-bit), 乱数 R (192-bit)
Output ビット列 m' (1024-bit)

1. $r =$ most significant 128-bit of $h(R)$.
2. $M = (m||0^{128}) \oplus G_2(r)$
3. $m' = M||(r \oplus H_2(M))$
4. Return m' and end.

1.7.5 $\mathrm{convert}^{-1}$(修正版)

Input ビット列 m' (1024-bit)
Output ビット列のペア (m, w); m は 768-bit, w は 192-bit.

1. $E =$ most significant 896-bit of m'.
2. $F =$ least significant 128-bit of m'. $\quad m' = E||F$
3. $W = G_2(H_2(E) \oplus F) \oplus E$
4. $m =$ most significant 768-bit of W.
5. $w =$ leastt significant 128-bit of W. $\quad W = m||w$
6. Return (m, w) and end.

1.7.6 decrypt(追加)

Input ビット列 x_1, x_2, x_3, x_4 (256-bit), 素数 p_1, p_2, p_3, p_4 (256-bit), 補助パラメータ z_1, z_2 (256-bit), z_3 (512-bit);
ただし $z_1 = p_2^{-1} \bmod p_1$, $z_2 = p_4^{-1} \bmod p_3$, $z_3 = (p_3 p_4)^{-1} \bmod (p_1 p_2)$
Output ビット列のペア (m, w); m は 768-bit, w は 192-bit.

1. $M_1 = \mathrm{PHI}(x_1, x_2, p_1, p_2, z_1)$
2. $M_2 = \mathrm{PHI}(x_3, x_4, p_3, p_4, z_2)$
3. $M = \mathrm{PHI}(M_1, M_2, (p_1 p_2), (p_3 p_4), z_3)$
4. return($\mathrm{convert}^{-1}(M)$) and end.

1.7.7 G_2(修正)

Input ビット列 x (128-bit)
Output ビット列 X (896-bit)

1. For $i = 1, 2, 3, 4, 5, 6, 7$, calculate $X_i = h'(x||C_i)$.
2. $X = X_1||X_2||X_3||X_4||X_5||X_6||X_7$.
3. return X and end.

注意 定数 $C_1, \ldots, C_7$ は別に設定されているとする.

1.7.8 H_2(修正)

Input ビット列 y (896-bit)
Output ビット列 Y (128-bit)

1. Divide $y = y_1||y_2||y_3||y_4||y_5||y_6||y_7$ ($|y_i| = 128$).
2. For $i = 1, 2, 3, 4, 5, 6, 7$, calculate $Y_i = h'(y_i||C_i)$.
3. $Y = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5 \oplus Y_6 \oplus Y_7$.
4. return Y and end.

注意 定数 $C_1, \ldots, C_7$ は別に設定されているとする.

1.7.9 h'(追加)

Input ビット列 x (256-bit)
Output ビット列 X (128-bit)

1. $X = h((x||x) \oplus C)$ の上位 128-bit
2. return X and end.

注意 定数 $C, C_1, \ldots, C_7$ と, 任意長の入力値に対して 160 ビット値を出力するハッシュ関数 h は別に設定されているとする.

2　攻撃評価

本節では, 前節で提示した HIME-2 修正アルゴリズムに基づいて, 暗号スキームと暗号プリミティブの両面から安全性評価を行う. 2.1 節では暗号スキームに対する安全性, すなわち HIME-2 の証明可能安全性について論じる. 評価書では HIME-2 は IND-CCA2 を達成していると主張されているが, 証明に不備が見受けられるため, そのような安全性は主張できないこと, すなわち, 評価書の証明では HIME-2 が IND-CCA2 の安全性を持つことを説明できないことを指摘する. 2.2 節では HIME-2 の暗号プリミティブの安全性, つまり素因数分解アルゴリズムへの耐性について議論する. 結論として HIME-2 の現在における安全性には問題が見られないものの, 鍵長などに可変性がないため, 将来的な安全性については, 近い将来には解読可能になり得ることを指摘する. 2.3 節では補助的に使用される関数 H_2 の安全性について議論する. この関数は通常のハッシュ関数よりもビット長が短いため, コリジョンが見つけやすくなってしまうという問題点を持つことを指摘する.

2.1　暗号スキームに対する安全性評価

HIME-2 の暗号スキームの持つ安全性に付いては, 以下の議論がなされている.

評価書 1.1 節 HIME-2 の安全性

まず, 次の定義を与える.

定義 1.1. $\mathcal{G}_0$ を, $\mathcal{G}_0(1^k) \to n,\ n = \prod_{i=1}^{d} p_i$, p_i $(1 \leq i \leq d)$ は素数, $|n| = k$, なる確率的インスタンス生成機とする. 但し, $\mathcal{G}_0$ の出力 n の分布は, HIME-2 における n の確率分布とおなじものとする.

素因数分解問題とは, 与えられた (n, d) に対して, p_i $(1 \leq i \leq d)$ を求める問題である. 素因数分解が困難であるとは, 任意の確率的多項式時間アルゴリズム A, 任意の定数 c, 十分大きな k に対して

$$\Pr(A(1^k, n, d) = (p_1, p_2, \ldots, p_d) | n \xleftarrow{R} \mathcal{G}_0(1^k)) < \frac{1}{k^c}.$$

が成立することである.

次の定義は, ランダムオラクルモデル上 (G, H が理想的ランダム関数) で適応的選択暗号文攻撃に対して強秘匿であることの定義を与える.

定義 1.2. $\mathcal{G}_1$ を, $\mathcal{G}_1(1^k) \to (pk, sk)$, pk は HIME-2 における公開鍵, sk は HIME-2 における秘密鍵, なる確率的鍵生成器とする. さらに, $\mathcal{E}_1$ を, $\mathcal{E}_1(pk, m) \to C$, m は平文, なる (HIME-2 における) 暗号文生成器とする.

HIME-2 が適応的選択暗号文攻撃に対して強秘匿 (IND-CCA2) であるとは, 任意の確率的多項式時間アルゴリズム Adv, M, 任意の定数 c, 十分大きな k, に対して,

$$\Pr(Adv^{D_{sk},G,H}(1^k, pk, m_0, m_1, C) = m | G, H \xleftarrow{R} \Omega; (pk, sk) \xleftarrow{R} \mathcal{G}_1(1^k);$$
$$(m_0, m_1) \xleftarrow{R} M^{D_{sk},G,H}; m \xleftarrow{R} \{m_0, m_1\}; C \xleftarrow{R} \mathcal{E}_1^{G,H}(pk, m)) < \frac{1}{2} + \frac{1}{k^c},$$

が成立することである. 但し Ω は $\{0,1\}^*$ から $\{0,1\}^\infty$ への写像全体を表す.

定義 1.2 において, 無視できないアドバンテージで正しい m を言い当てる確率的多項式時間アルゴリズム Adv が存在したとすると, Adv は,

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

を無視できない確率で計算できることが文献 [1] に示されている. よって, Adv は暗号文 C に対して, 2^d 個存在する $\sqrt{C}$ の少なくとも 1 つを無視できない確率で計算できる能力を持つ. 以上のことから, このような Adv を利用して, n の素因数分解を行う確率的多項式時間アルゴリズムを容易に構成することができる.

よって, 次の定理が導かれる.

定理 1.1 (IND-CCA2). G, H を理想的ランダム関数とする. このとき, HIME-2 は, 素因数分解問題の困難性を前提に, 適応的選択暗号文攻撃に対して強秘匿である.

OAEP の安全性に関しては

> **定理 (Bellare-Rogaway,[評 1])** ランダムオラクルモデルの下で (すなわちランダム関数 G, H が理想的にランダムであるという仮定の下で), 一方向性置換 f が OW-CPA であるとき, OAEP は IND-CCA である.

が成立するので, HIME-2 の安全性に対しては, HIME-2 の暗号化関数が OAEP を適用可能か, という 1 点に集約できる. ここで HIME-2 の暗号化関数 f_{HIME2} とは, 与えられた合成数 n と $x \in \mathbf{Z}_n^*$ に対して $f_{HIME2}(x) = x^2 \bmod n$ である. 以下, f_{HIME2} への OAEP の適用可能性について議論する.

2.1.1 一方向性関数であること

f_{HIME} が一方向性であるとは, f_{HIME} の逆関数の計算が計算量的に困難であることを意味し, 具体的には $m^2 \bmod n$ から $m \bmod n$ を計算する問題である. この問題の困難性に関しては, n の素因数分解問題と計算量的な同等性が知られている (例えば [評 14] の 7.4.3 節など). HIME-2 では素因数分解問題の困難性が仮定されているので, 関数 f_{HIME2} は一方向性を有していることが示される.

ただし以上の議論は自明でないため, HIME-2 の暗号化関数の一方向性を主張する上で何らかの議論が必要である. 特に暗号の安全性の自己評価を行うにおいては, 最低でも参考文献の引用が必要であろう. この意味で, 自己評価書では大きく情報が欠けていると捉えることができる.

2.1.2 置換であること

HIME-2 の暗号化関数 f_{HIME2} は Rabin 暗号の暗号化関数と同じであり, [評 1] において OAEP の Rabin 暗号への適用可能性が記述されていることから, 評価書では f_{HIME2} が置換であることが暗黙の了解となっている.

しかし簡単な議論から, HIME-2 の暗号化関数は置換でない ことを示すことができる. 関数 f_{HIME2} は与えられた値の平方を計算する関数だから, その値域は $\mathbf{Z}_n^*$ の平方剰余な元全体 QR である. よって f_{HIME2} が置換であるためには, その定義域も QR に一致する必要がある. しかし仕様書 3.2.2 節の暗号化アルゴリズムではそのような制限を行っておらず, 関数の定義域は $\mathbf{Z}_n^*$ 全体となっている. このため暗号化関数 f_{HIME2} は $\mathbf{Z}_n^*$ 上の 2:1 の写像であって, 置換にはなっていない.

以上の議論から, HIME-2 の暗号化関数 f_{HIME2} は OAEP の仮定を満たしていないため, 評価書に記載された定理 1.1 の主張は誤りである. よって HIME-2 は証明可能安全性を有していないと結論づけることができる.

2.1.3 OAEP の安全性

暗号化アルゴリズムを修正することによって, 暗号化関数 f_{HIME2} が置換性を持ったとしても, HIME-2 が IND-CCA2 の安全性を達成するには, 評価書の記載だけでは不十分である.

証明可能安全性に関しては, Bellare-Rogaway の定理を理由にして

> **予想** ランダムオラクルモデルの下で (すなわちランダム関数 G, H が理想的にランダムであるという仮定の下で), 一方向性置換 f が OW-CPA であるとき, OAEP は IND-CCA2 である.

であることが長い間当然視されてきた.[5] しかしごく最近になって, この予想が誤りであることが Shoup によって指摘された ([Sho00]). Fujisaki らの結果によると, この予想は以下のようしてに修正されることも報告されている ([FOPS00]).

> **定理 (Fujisaki et al.,[FOPS00])** ランダムオラクルモデルの下で (すなわちランダム関数 G, H が理想的にランダムであるという仮定の下で), 一方向性置換 f が Partial OW-CPA であるとき, OAEP は IND-CCA2 である.

これら OAEP の安全性に関する議論の経緯から考えると, 評価書が主張する [評 1] だけの引用では不十分であり, HIME-2 が IND-CCA2 であることを証明するには, 暗号化関数 f_{HIME2} が Partial OW であることを示す必要がある. しかしその証明は与えられていないため, HIME-2 が IND-CCA2 であることを主張することは困難である.

以上まとめると, HIME-2 は暗号化関数が置換性を有していないため, 証明可能安全性の仮定を満たしておらず, その結果として証明可能安全性を有していない. また, 置換性を有するように修正できたとしても, その関数

[5] これは CCA1 と CCA2 の概念が明確に区別されていなかったことが原因と考えられる.

が Partial One-Wayness を持つことが示されない限り, HIME-2 は IND-CCA2 を達成しえないことが結論づけられる.

2.2 暗号プリミティブに対する安全性評価

本節では, HIME-2 の暗号プリミティブの安全性評価として, 素因数分解問題の困難性に関する議論を行う. またその結果をもとに, HIME-2 の採用するパラメータの正当性についての検討も行う.

結論として, HIME-2 の鍵長は現時点では安全と思われるものの, 鍵長などに可変性がないため, 今後数年のうちに解読可能になり得ることを指摘する. また鍵長などのパラメータの設定理由が明示されておらず, 特に d が 4 に設定されている理由が明確でない点に疑問を呈する.

2.2.1 素因数分解アルゴリズムに対する耐性

素因数分解アルゴリズムの現状と HIME-2 の耐性については, 評価書において以下の議論が行われている.

評価書 1.2 節 素因数分解の安全性について
素因数分解アルゴリズムの計算量は合成数 n のサイズのみによって決まる合成数依存型, n の素因数のサイズによって決まる素因数依存型の 2 つに分けることが出来る. 素因数依存型のアルゴリズムは ρ 法, $p-1$ 法, $p+1$ 法, 楕円曲線法 ([9],[12]) などが知られており, 合成数依存型としては (一般) 数体ふるい法 ([10]) が強力である.
HIME-2 は, 256-bit の素数 $p_i(i=1,2,3,4)$ を素因数にもつ 1024-bit の合成数 $n=p_1p_2p_3p_4$ を用いる. このサイズに対し, ρ 法の分解能力は低く, また, $p-1$ 法, $p+1$ 法に対しては素数 (秘密鍵) の生成時に適切な ($p-1, p+1$ が十分大きな素因数を持つ) ものを取ることで対応している. (暗号技術仕様書参照).
楕円曲線法は n の素因数 p を求める平均的 (かつ漸近的) な計算量が $L_p[1/2,\sqrt{2}]$ ($L_p[a,b] = \exp\left((b+o(1))(\log p)^a(\log\log p)^{1-a}\right)$) であり, 数体ふるい法では理論的には $L_n[1/3, 1.901]$ ([3]) などの見積もりが算出されており, いずれも入力サイズの準指数関数となっている. 実際には分解アルゴリズムの実装や現在の計算機能力などの事情により素因数分解が成功しているのは, 楕円曲線法では素因数のサイズが 180 ~ 190-bit 程度, また数体ふるい法では合成数のサイズが 512-bit 程度である ([4],[16] 参照).
これらの現状を考慮すると現在知られている素因数分解方法により提案法式で用いる合成数を有効に分解することは困難であると考えられる.

素因数分解アルゴリズムに関する上記記述は, 現状の認識として十分不可欠であると考える. よって HIME-2 のしている 1024-bit の合成数および 256-bit の素因数は (現在の) 素因数分解アルゴリズムに対して十分な耐性を持つと結論づけられる.

2.2.2 将来的な耐性

次に HIME-2 のパラメータの将来的な安全性について考える. 量子計算機の実用化などによって, 多項式時間で計算可能な素因数分解アルゴリズムの実現性は否定できないが, そのような実現性を仮定することは極めて困難である. 本節では計算機能力の進歩だけを仮定した場合に, 素因数分解アルゴリズムの分解能力がどの程度進展するかを考察することによって, HIME-2 の暗号プリミティブの安全性を議論する.

素因数分解アルゴリズムのうち, (一般) 数体ふるい法は最良のアルゴリズムである. 2000 年 12 月時点での世界記録は 512-bit であり ([RSA-155], [6] その計算時間は 7 ヶ月, 計算資源で 8400 MIPS Year と報告されている. 数

[6] 2000 年 11 月に特殊数体ふるい法が 774-bit の分解に成功した ([SNFS-233]) が, これは n が特殊な場合であり, 一般の場合と同様には

体ふるい法の計算量から考えると, 1024-bit の素因数分解に必要な計算量は 512-bit の場合の約 7500000 倍となる. したがって [RSA-155] と同じ計算機資源を用いた場合, 分解に必要な時間は約 4400000 年となり, 事実上分解不可能である. 数体ふるい法の途中の部分を専用ハードウェアで高速化することが可能であるという報告 [LS00] もあるが, それを加味しても分解に必要な時間は約 500000 年となる試算 [Sil99] があり, 事実上分解不可能であることには変わりない. ただしこれらの試算には計算能力の向上が反映されていない.

計算能力を加味した試算として, [RSA-155] には数体ふるい法による記録 (桁数 D 桁) と達成年 (西暦 Y) の間の近似式

$$Y = 13.24D^{1/3} + 1928.6$$

が紹介されており, この近似式からは 1024-bit (309 桁) の素因数分解は 2018 年に可能と導かれる. 近似式は経験的な結果であり, 信憑性には議論の余地があるが, [RSA-155] から類推される結果を高く信用できないことを示唆しているとも考えられる.

なお [LV00] では, 各種暗号の安全な鍵長が時間軸にそった形で提示されている. この報告によると, RSA 暗号の鍵長の 2002 年での推奨値は 1028-bit となっている. HIME-2 にも類推が可能であり, この報告の立場をとる限り, 2003 年以降は HIME-2 を推奨できないことになることも指摘しておく.

以上の議論は合成数の大きさに関連した議論であり, HIME-2 ではこの他に素因数の大きさについての試算も必要である. この場合, 楕円曲線法がその対象となるが, 残念ながら定量的な見積もり報告はなされていない. ただし数体ふるい法と同様に, Brent は楕円曲線法による記録 (桁数 D 桁) と達成年 (西暦 Y) の間の近似式

$$Y = 9.3\sqrt{D} + 1932.3$$

を与えている ([Bre00]). この近似式から 256-bit(76.8 桁) の素因数分解は 2013 年に可能という結果を得られる.

以上まとめると, HIME-2 の素因数分解アルゴリズムに対する耐性は, 現時点では問題がないと考えられる. また将来的な耐性でも, 数年のスパンでは問題がないが, 2015 〜 2020 年頃には分解可能という試算もあり, 長期的な安全性は保証できない.

2.3 補助関数 H_2 の安全性

HIME-2 では補助関数として関数 H_2 を使用しており, 仕様書ではこの関数を "ハッシュ関数" と呼んでいる. しかし H_2 のハッシュ性に関する議論は何もなされておらず, また本稿 1.1.1 節で指摘したとおりその仕様には不備が見られるため, そもそも H_2 のハッシュ性は安全性評価のためには欠かせないと考える.

しかし H_2 がハッシュ性を持ったとしても, この関数の安全性の問題点を指摘できる. 仕様によると H_2 の入力長は任意, 出力長は 128-bit となっている. したがって H_2 のコリジョンは 2^{64} 程度の計算量で求めることが可能である. この計算が現実的かどうかに関する議論は報告されていないが, この計算量を 64-bit 鍵長の共通鍵暗号の安全性と同等と見なし, [LV00] の報告を引用すれば, これは 682-bit の鍵長を持つ RSA 暗号の解読に必要な計算量と同じである. 本稿 2.2.2 節と同様な計算量比較によれば, これは [RSA-155] の約 400 倍でしかない. したがって HIME-2 では関数 H_2 のコリジョンを計算可能であると結論づけられる.

論じられない.

3　パラメータの正当性

本節では HIME-2 で使用されているパラメータの設定値の正当性について議論する．仕様書ではパラメータ $|n|, k_0, k_1, d$ が天下り的に定められており, 設定理由に関しては明記されていない．このことは暗号設計の透明性と関連することから, 妥当性についての議論は不可欠であり, このような記述が見られないと言う点では, 評価書の記述内容は不十分であると捉えることができる．

以上の理由から HIME-2 の安全性を議論する上では, 設定値の正当性を論じる必要である．結論として $|n|$ については妥当と思われるが, 解読に必要な計算量などの観点から考えて, その設定理由を明示することが望ましいと考える．また k_0, k_1, d については設定理由が不明瞭であり, 必ずしも妥当な設定値とは断定できない．

3.1　鍵長 $|n|$ について

鍵長 $|n|$ の設定値については, 仕様書 3.2 節に「実際の使用を想定して, 鍵長を 1024 ビット ($|n| = 1024$) とし....」との記述があるが, この値の妥当性については一切説明されていない．

確かに, 標準的に使用されている RSA 暗号と HIME-2 の暗号プリミティブは, どちらも安全性の根拠を素因数分解問題の困難性に依存しており, RSA 暗号の鍵の推奨値が 1024-bit (以上) といわれていることから HIME-2 の鍵長も 1024-bit 以上にすべきとの推論は可能である．しかし HIME-2 が使用する素因数の大きさは 256-bit であって, 鍵長 1024-bit の RSA 暗号の 512-bit に比べれば, 安全性が低下していることは否めない．

3.2　k_0, k_1 について

k_0, k_1 については, 仕様書 3.2.6 節の記述から $k_0 = k_1 = 128$ であることがわかるが, これら値の妥当性についても一切説明されていない．

仕様書 2.1 節に「HIME-2 は Rabin 暗号を OAEP の手法により変換した方式に改良を加えたものである.」との記述があり, OAEP の手法から大きく影響を受けていると考えられる．しかし OAEP の手法を説明した論文 [仕 4] の記載を見ると, 例として挙げられている RSA-OAEP では $k_0 = k_1 = 160$ という値が使用されており, HIME-2 がそれよりも小さい値を採用した理由は推測できない．[7] しかし本稿 2.3 節で指摘した通り, $k_0 = k_1 = 128$ とした場合にはハッシュ関数のコリジョンが計算しやすくなるため, 安全性を犠牲にしてしまっている．

3.3　d について

d の具体的な値については, 仕様書 3.2 節 $d = 4$ と設定されているが, この値の妥当性についても一切説明されていない．

そもそも暗号技術概要説明書の 2.1 設計方針の項においては, $d \leq 3$ の場合でも HIME-2 と同様なアルゴリズムは構成可能である．確かに仕様書 2.2.4 節では $d = 4 \sim 6$ が推奨されているものの, 暗号プリミティブに対する楕円曲線法を用いた素因数分解アルゴリズムの適用を考えた場合, $d = 6$ のとき $|p_i| = 171$ となり n は分解可能であるし, $d = 5$ のときでも $|p_i| = 205$ となり分解可能性は決して高いとはいえない．したがって d は小さい方が望ましいことは導かれる．

それならばどうして $d = 3$ を採用しなかったのかという疑問が残る．この場合 $|p_i| = 342$ となり, 素因数分解からの脅威は $d = 4$ のときよりもはるかに小さくなるはずである．安全性を犠牲にしてまで $d = 4$ を採用した理由が推測できない．

[7] HIME-2 の暗号技術概要説明書の 2.1 設計方針 の項において「平文空間を広く選ぶように設計した」との記述があるので, これが k_0, k_1 の値を [仕 4] の例よりも小さくした理由と邪推することも可能である．

4 まとめ

本稿では, まずはじめに HIME-2 の技術仕様書にて定義されるアルゴリズムの問題点を指摘し, 安全性評価が可能となるように, 修正アルゴリズムを提示した. 次に修正アルゴリズムに基づいて, 暗号スキームと暗号プリミティブの各レベルにおける安全性評価を行った. 暗号スキームレベルでは, 評価書には HIME-2 は証明可能安全性を持つとの主張があるものの, 2 つの理由により, その主張の正当性は低いと考える. また暗号プリミティブのレベルでは, 現在での安全性には問題ないものの, 将来的な安全性については, 鍵長などに可変性がないため, 今後数年のうちに解読可能になり得ることを指摘した. さらには HIME-2 のパラメータの設定値の正当性について論じ, 安全性が犠牲になっている現象が見られることから, 設定理由の必要性を指摘した.

本稿の参考文献は以下の通りである. [評 1] とある場合には, HIME-2 の自己評価書の参考文献 [1] を意味するものとする.

参考文献

[Bre00] R. P. Brent, *Recent progress and prospects for integer factorisation algorithms*, To be appear in proceedings of COCOON 2000 (Sydney, July, 2000). Available from Brent's website `http://web.comlab.ox.ac.uk/oucl/work/richard.brent/pub/pub196.html`.

[FOPS00] E. Fujisaki, T. Okamoto, D. Pointcheval, J. Stern, *RSA-OAEP is Still Alive!*, preprint. Available from `http://eprint.iacr.org/2000/061.pdf`

[LS00] A. K. Lenstra, A. Shamir, *Analysis and Optimization of the TWINKLE Factoring Device*, proceedings of EUROCRYPTO2000 (Bruges, May, 2000), LNCS 1807, 2000.

[LV00] A. K. Lenstra, E. R. Verheul, *Selecting Cryptographic Key Sizes*, proceedings of PKC2000 (Merborne, January, 2000), LNCS 1751, 2000.

[RSA-155] S. Cavallar et al., *Factorization of a 512-Bit RSA Modulus*, proceedings of EUROCRYPTO2000 (Bruges, May, 2000), LNCS 1807, 2000.

[Sho00] V. Shoup, *OAEP Reconsidered*, preprint. Available from `http://eprint.iacr.org/2000/060.pdf`

[Sil99] R. D. Silverman, *An Analysis of Shamir's Factoring Device*, RSA Laboratories' Bulletin, No.12, May 3, 1999. Available from `http://www.rsasecurity.com/rsalabs/bulletins/`

[SNFS-233] Internet announcement. Available from `ftp://ftp.cwi.nl/pub/herman/SNFSrecords/SNFS-233/`