暗号アルゴリズムの評価

2001 年度

神 戸 大 学

桑門 秀典

暗号アルゴリズムの評価

桑門 秀典

神戸大学工学部電気電子工学科

目 次

1 序論

Draft FIPS-180-2[11] では，256bits, 384bits, 512bits のハッシュ値を出力するハッシュ関数 SHA-256, SHA-384, SHA-512 が提案されている．現在広く使用されている SHA-1[10] の安全性が損なわれるような報告はないが，SHA-1 のハッシュ値は 160bits なので，計算機の性能向上によってその安全性が脅かされる可能性も否定できない．

SHA-256, 384, 512 は，衝突困難かつ一方向性であるように設計されている．また，同時に入力メッセージの少しの変化が出力のハッシュ値に大きく影響するように設計されている．Draft FIPS-180-2 でも述べられているように，SHA-256, 384, 512 をディジタル署名，鍵付ハッシュによるメッセージ認証，擬似乱数生成器として使用するためには，これらの設計目標が達成されていなければならない．これらの設計目標が達成されていないという証拠はないが，逆に設計目標が達成されているという数学的な証明もないという現状に注意する必要がある．

SHA-256, 384, 512 の構造は，SHA-1 と同様に Merkle-Damgård が提案した衝突困難なハッシュ関数の構成法に準じている [5]．メッセージをいくつかのブロックに分割し，各ブロックを圧縮関数 f により繰り返し処理をする構造になっている (図 1)．この構成法の特徴は，ハッシュ関数全体の衝突困難性が圧縮関数 f の衝突困難性に帰着されることである．

一方，Merkle-Damgård の構成法に基づくハッシュ関数は，必ずしも汎用一方向性ハッシュ関数ではないことに注意する必要がある [2]．汎用一方向性ハッシュ関数を構成する方法として，Shoup の構成法が知られているが [13]，SHA-256, 384, 512 は Shoup の構成法に準じていない．

本報告書では，SHA-256, 384, 512 の安全性を評価する．ただし，SHA-384 は，SHA-512 と本質的に同じなので，主に SHA-256, 512 について論じる．本報告書の構成は，以下のとおりである．

2.1 章: SHA-256, 512 のアルゴリズムと記号の定義を述べる．

2.2 章: SHA-1 と SHA-256, 512 のアルゴリズムの相違点をまとめる．

2.3 章: Draft FIPS-180-2 への公開コメントを簡単にまとめる．

3.1 章: SHA-256, 512 の安全性の評価基準について論じている．

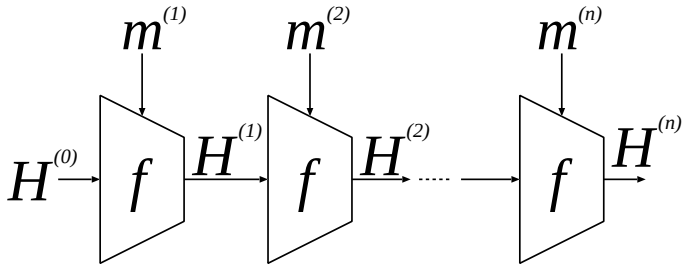

図 1: Merkle-Damgård の構成法

3.2 章: SHA-256, 512 において，入力のビット系列の変化が出力のビット系列に及ぼす影響を実験的に調べている．

3.3 章: birthday paradox を利用した攻撃法を実際に行った結果を述べている．

3.4 章: SHA-256, 512 のアルゴリズム的並列処理の可能性とそれが安全性に及ぼす影響を述べている．

3.5 章: 他のハッシュ関数への攻撃法を SHA-256, 512 に適用できるかどうかを検討している．

3.6 章: SHA-256, 512 の一方向性について検討している．

3.7 章: Kelsey が提案している SHA-256, 512 の改良法の評価を行っている．

4 章: 本報告書のまとめを述べている．

2 SHA-256, 512

2.1 アルゴリズム

この章では，Draft FIPS-180-2 の SHA-256, 512 のアルゴリズムを述べる．Draft FIPS-180-2 で記述されているアルゴリズムと本質的に同じであるが，後の安全性解析の記述を明確にするため，アルゴリズム中の変数等の表記法が Draft FIPS-180-2 と異っている．

2.1.1 関数の定義

SHA-256, 512 で共通に使用される関数をまとめる．変数のビット長は，SHA-256 の場合は 32bits，SHA-512 の場合は 64bits である．

$$\mathrm{Ch}(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$\mathrm{Maj}(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$
$\mathrm{SHR}^n(x) : x$ を n bits 右へシフト
$\mathrm{ROTR}^n(x) : x$ を n bits 右へ巡回シフト

2.1.2 SHA-256

メッセージ m を padding して，512 の倍数の長さのビット系列に変換する．padding 方法については，Draft FIPS-180-2 p. 12 を参照．そして，m を n 個の 512bits のメッセージブロックに分割する．それを $m^{(1)}, m^{(2)}, \ldots, m^{(n)}$ と書く．$i = 1, 2, \ldots, n$ に対して，以下の処理を行う (各変数は 32bits)．メッセージ m のハッシュ値は，$H_0^{(n)}|H_1^{(n)}|\cdots|H_7^{(n)}$ である．

1. $m^{(i)}$ を 32bits 毎に分割し，それらを $m[t]$ $(0 \leq t \leq 15)$ と書く．そして，$w[t]$ を以下のように計算する．

$$w[t] = \begin{cases} m[t] & 0 \leq t \leq 15 \\ \sigma_1^{\{256\}}(w[t-2]) + w[t-7] + \sigma_0^{\{256\}}(w[t-15]) + w[t-16] & 16 \leq t \leq 63 \end{cases} \tag{1}$$

ここで，

$$\sigma_0^{\{256\}}(x) = \mathrm{ROTR}^7(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{SHR}^3(x) \tag{2}$$
$$\sigma_1^{\{256\}}(x) = \mathrm{ROTR}^{17}(x) \oplus \mathrm{ROTR}^{19}(x) \oplus \mathrm{SHR}^{10}(x) \tag{3}$$

である．

2. 変数を初期化する[1]．

$$\begin{aligned} a[0] &= H_0^{(i-1)}, \quad b[0] = H_1^{(i-1)}, \\ c[0] &= H_2^{(i-1)}, \quad d[0] = H_3^{(i-1)}, \\ e[0] &= H_4^{(i-1)}, \quad f[0] = H_5^{(i-1)}, \\ g[0] &= H_6^{(i-1)}, \quad h[0] = H_7^{(i-1)} \end{aligned}$$

なお，$H_0^{(0)}, H_1^{(0)}, \ldots, H_7^{(0)}$ については，Draft FIPS-180-2 p. 13 を参照．

3. $t = 0, 1, \ldots, 63$ に対して以下の計算を繰り返す (一つの t に対する以下の処理をまとめて，1 ステップと呼ぶ)．

$$T_1[t] = h[t] + \Sigma_1^{\{256\}}(e[t]) + \mathrm{Ch}(e[t], f[t], g[t]) + K_t^{\{256\}} + w[t] \tag{4}$$
$$T_2[t] = \Sigma_0^{\{256\}}(a[t]) + \mathrm{Maj}(a[t], b[t], c[t]) \tag{5}$$
$$a[t+1] = T_1[t] + T_2[t] \tag{6}$$
$$b[t+1] = a[t] \tag{7}$$
$$c[t+1] = b[t] \tag{8}$$
$$d[t+1] = c[t] \tag{9}$$
$$e[t+1] = d[t] + T_1[t] \tag{10}$$
$$f[t+1] = e[t] \tag{11}$$
$$g[t+1] = f[t] \tag{12}$$
$$h[t+1] = g[t] \tag{13}$$

ここで，

$$\Sigma_0^{\{256\}}(x) = \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x) \tag{14}$$
$$\Sigma_1^{\{256\}}(x) = \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x) \tag{15}$$

である．なお，$K_t^{\{256\}}$ については，Draft FIPS-180-2 p. 9 を参照．

[1] Draft FIPS-180-2 では，$H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}$ で初期化している．これは Draft FIPS-180-2 の誤記と思われる．SHA-1, 384, 512 の記述にも同様の誤記がある．

4. i 番目の中間ハッシュ値を計算する．

$$\begin{aligned} H_0^{(i)} &= a[64] + H_0^{(i-1)}, \quad H_1^{(i)} = b[64] + H_1^{(i-1)}, \\ H_2^{(i)} &= c[64] + H_2^{(i-1)}, \quad H_3^{(i)} = d[64] + H_3^{(i-1)}, \\ H_4^{(i)} &= e[64] + H_4^{(i-1)}, \quad H_5^{(i)} = f[64] + H_5^{(i-1)}, \\ H_6^{(i)} &= g[64] + H_6^{(i-1)}, \quad H_7^{(i)} = h[64] + H_7^{(i-1)} \end{aligned}$$

上記の 1.∼4. までの処理が SHA-256 の圧縮関数である．

2.1.3 SHA-512

メッセージ m を padding して，1024 の倍数の長さのビット系列に変換する．padding 方法については，Draft FIPS-180-2 p. 12 を参照．そして，m を n 個の 1024bits のメッセージブロックに分割する．それを $m^{(1)}, m^{(2)}, \ldots, m^{(n)}$ と書く．$i = 1, 2, \ldots, n$ に対して，以下の処理を行う (各変数は 64bits)．メッセージ m のハッシュ値は，$H_0^{(n)}|H_1^{(n)}|\cdots|H_7^{(n)}$ である．

1. $m^{(i)}$ を 64bits 毎に分割し，それらを $m[t]$ $(0 \leq t \leq 15)$ と書く．そして，$w[t]$ を以下のように計算する．

$$w[t] = \begin{cases} m[t] & 0 \leq t \leq 15 \\ \sigma_1^{\{512\}}(w[t-2]) + w[t-7] + \sigma_0^{\{512\}}(w[t-15]) + w[t-16] & 16 \leq t \leq 79 \end{cases} \tag{16}$$

ここで，

$$\sigma_0^{\{512\}}(x) = \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x) \tag{17}$$

$$\sigma_1^{\{512\}}(x) = \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x) \tag{18}$$

である．

2. 変数を初期化する．

$$\begin{aligned} a[0] &= H_0^{(i-1)}, \quad b[0] = H_1^{(i-1)}, \\ c[0] &= H_2^{(i-1)}, \quad d[0] = H_3^{(i-1)}, \\ e[0] &= H_4^{(i-1)}, \quad f[0] = H_5^{(i-1)}, \\ g[0] &= H_6^{(i-1)}, \quad h[0] = H_7^{(i-1)} \end{aligned}$$

なお，$H_0^{(0)}, H_1^{(0)}, \ldots, H_7^{(0)}$ については，Draft FIPS-180-2 p. 14 を参照．

3. $t = 0, 1, \ldots, 63$ に対して以下の計算を繰り返す (一つの t に対する以下の処理をまとめて，1 ステップと呼ぶ)．

$$T_1[t] = h[t] + \Sigma_1^{\{512\}}(e[t]) + \mathrm{Ch}(e[t], f[t], g[t]) + K_t^{\{512\}} + w[t] \tag{19}$$

$$T_2[t] = \Sigma_0^{\{512\}}(a[t]) + \mathrm{Maj}(a[t], b[t], c[t]) \tag{20}$$

$$a[t+1] = T_1[t] + T_2[t] \tag{21}$$

$$
\begin{aligned}
b[t+1] &= a[t] && (22)\\
c[t+1] &= b[t] && (23)\\
d[t+1] &= c[t] && (24)\\
e[t+1] &= d[t] + T_1[t] && (25)\\
f[t+1] &= e[t] && (26)\\
g[t+1] &= f[t] && (27)\\
h[t+1] &= g[t] && (28)
\end{aligned}
$$

ここで,

$$
\begin{aligned}
\Sigma_0^{\{512\}}(x) &= \mathrm{ROTR}^{28}(x) \oplus \mathrm{ROTR}^{34}(x) \oplus \mathrm{ROTR}^{39}(x) && (29)\\
\Sigma_1^{\{512\}}(x) &= \mathrm{ROTR}^{14}(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{ROTR}^{41}(x) && (30)
\end{aligned}
$$

である．なお，$K_t^{\{512\}}$ については，Draft FIPS-180-2 p. 10 を参照.

4. i 番目の中間ハッシュ値を計算する．

$$
\begin{aligned}
H_0^{(i)} &= a[80] + H_0^{(i-1)}, \quad H_1^{(i)} = b[80] + H_1^{(i-1)},\\
H_2^{(i)} &= c[80] + H_2^{(i-1)}, \quad H_3^{(i)} = d[80] + H_3^{(i-1)},\\
H_4^{(i)} &= e[80] + H_4^{(i-1)}, \quad H_5^{(i)} = f[80] + H_5^{(i-1)},\\
H_6^{(i)} &= g[80] + H_6^{(i-1)}, \quad H_7^{(i)} = h[80] + H_7^{(i-1)}
\end{aligned}
$$

上記の 1.∼4. までの処理が SHA-512 の圧縮関数である．

SHA-384 のアルゴリズムは，以下の二点を除いて SHA-512 のアルゴリズムと同じである．

- 2. の $H_i^{(0)}$ の値が異る．
- メッセージ m のハッシュ値を $H_0^{(n)}|H_1^{(n)}|\cdots|H_5^{(n)}$ とする．

2.2 SHA-256, 512 の特徴

SHA-256, 512 のアルゴリズムは，SHA-1 のアルゴリズムとかなり異る．その相違点を表 1 ∼ 表 3 にまとめる．

安全性に関係すると思われる変更点を列挙する．

- メッセージ拡張処理が複雑になっている．これは，後の章で述べるように，安全性の向上に役立っている．
- 1 ステップで更新される変数が二つになっている．これは，後の章で述べるように，既存の攻撃法の適用を困難にしている．

なお，SHA-1 では，巡回シフトが全て左向きであるの対し，SHA-256 では，全て右向きである．左向き巡回シフトは，右向き巡回シフトで実現することができるので，この変更は安全性に影響しない．

表 1: 入出力サイズと内部のステップ数

	ワード [bits]	メッセージ [bits]	ハッシュ値 [bits]	ステップ数
SHA-1	32	512	160	80
SHA-256	32	512	256	64
SHA-512	64	1024	512	80

表 2: 1 回のメッセージ拡張処理

	演算の種類	演算回数
SHA-1	$\oplus$, ROTL(),	4
SHA-256	$+$, $\oplus$, ROTR(), SHR()	13
SHA-512	$+$, $\oplus$, ROTR(), SHR()	13

表 3: 1 ステップの処理

	演算の種類	演算回数
SHA-1	$+$, ROTL(), $\wedge$, $\vee$, $\neg$, $\oplus$	8 又は 10
SHA-256	$+$, ROTR(), $\oplus$, $\wedge$, $\vee$, $\neg$	25
SHA-512	$+$, ROTR(), $\oplus$, $\wedge$, $\vee$, $\neg$	25

2.3 Draft FIPS-180-2 への公開コメント

Draft FIPS-180-2 に対しては，三つの公開コメントがある [12]．そのうちの一つは技術的内容がないので，それ以外の二つのコメントを以下にまとめる．

2.3.1 Jonsson のコメント

Jonsson のコメントの主旨を以下に述べる．いずれも妥当なコメントである．特に 1. は重要であり，NIST の今後の対応が注目される．

1. SHA-256, 384, 512 の設計は，SHA-1 の設計とはかなり異る．NIST は SHA-256, 384, 512 の安全性評価を公開し，SHA-1 と異る設計にした理由を明らかにせよ．
2. SHA-256 の出力を 160bits に制限した方法も SHA-1 の代替として標準化してはどうか．
3. SHA-256 と SHA-512 のアルゴリズムは非常に良く似ているため，コードサイズを小さくすることができることを明記すべきである．

2.3.2 Kelsey のコメント

Kelsey は，SHA-256, 384, 512 のもつ extension property について言及している．この extension property は，SHA-256, 384, 512 固有の性質ではなく，MD4 の改良と位置づけられるハッシュ関数全て (SHA-1, MD5 等) がもつ性質である．extension property が問題となるのは，SHA-256, 384, 512 を鍵付ハッシュ関数として利用する場合である．Kelsey は extension property 問題を解決する方法を提案している．

以下に Kelsey のコメント，特に extension property とその解決法についてまとめる．

Extension property SHA-256, 384, 512 を $h()$ と書く．あるメッセージ m に対するハッシュ値を $h(m)$ とする．このとき，m が未知であっても，m の長さと $h(m)$ から別のメッセージ $m|p|m'$ のハッシュ値 $h(m|p|m')$ を計算することができる．ここで，p は $h(m)$ を計算するときに m に付加される padding データ (主に m の長さ) であり，m' は任意のメッセージである．

Extension property 問題 送信者と受信者が鍵 k を共有しており，k を用いてメッセージ認証を行うとする．メッセージ m に対する MAC 関数を

$$MAC(k, m) = h(k|m)$$

とする．このとき，攻撃者は k の長さ, m の長さ及び $h(m)$ が分かれば，extension property より，別のメッセージ $m|p|m'$ の MAC 値を計算することができる．

解決方法 $h()$ のアルゴリズムを以下のように変更する．m の最終ブロック (padding データを含むブロック) を処理するとき，$H_0^{(n-1)}, H_1^{(n-1)}, \ldots, H_7^{(n-1)}$ に固定ビット系列 (0xa5a5$\cdots$a5) を排他的論理和で加える，つまり，

$$H_j^{(n-1)} = H_j^{(n-1)} \oplus \text{0xa5a5a5a5} \quad (j = 0, 1, \ldots, 7).$$

このように更新された $H_j^{(n-1)}$ に対して，最終ブロックの処理を行う．

3 安全性の検討

3.1 安全性の評価基準

Draft FIPS-180-2 では，ハッシュ関数の安全性を同じハッシュ値になる異るメッセージ (衝突) を発見するために必要な計算量で評価している．これは妥当な評価基準である．

これまでに代表的なハッシュ関数各々に対して安全性の検討，攻撃法の提案がなされている．しかし，強力な攻撃法であるほど，個々のハッシュ関数の仕様に強く依存しているため，その攻撃法を別のハッシュ関数の安全性の検討に利用することは難しい．どのハッシュ関数にも適用可能な攻撃法 (衝突を発見する方法) としては，birthday paradox に基づく攻撃法 (BP 攻撃法) 以外に見当たらない．

Draft FIPS-180-2 では，SHA-256, 384, 512 に対して，BP 攻撃法により衝突を発見するために必要なメッセージ数を各々$2^{128}, 2^{192}, 2^{256}$ と見積もっている．これは，SHA-256, 384, 512 が任意長のメッセージを 256, 384, 512bits に写像する理想的なランダム関数と仮定した場合，衝突するメッセージを確率約 1/2 で発見するために必要なメッセージ数である．

したがって，SHA-256, 384, 512 が安全であることを示すには，あらゆる攻撃法の計算量がこれらのメッセージ数を生成する計算量よりも多いことを示す必要がある．SHA-256, 384, 512 への攻撃方法は，主に次の四つに分類される．

ランダム性 暗号プロトコルでの利用を考えると，ハッシュ関数の出力は，擬似乱数のようになっていることが望ましい．例えば，入力ビット系列の少しの変化が全ての出力ビット系列に影響する方が良い．これについては，3.2 章で述べる．

BP 攻撃法 SHA-256, 384, 512 が任意長のメッセージを 256, 384, 512bits に写像する理想的なランダム関数でない場合，少ないメッセージ数で衝突が生じる可能性がある．これについては，3.3 章と 3.4 章で述べる．

衝突困難性 SHA-256, 384, 512 を $h()$ とするとき，$h(m) = h(m')$ となる二つの異るメッセージ m, m' を発見することが難しくなければならない．これについては，3.5 章で述べる．

一方向性 SHA-256, 384, 512 を $h()$ とする．一方向性を二つの場合に分けて考える．

1. m と $h(m)$ が与えれたとき，$h(m) = h(m')$ となる m' を発見することが難しい．

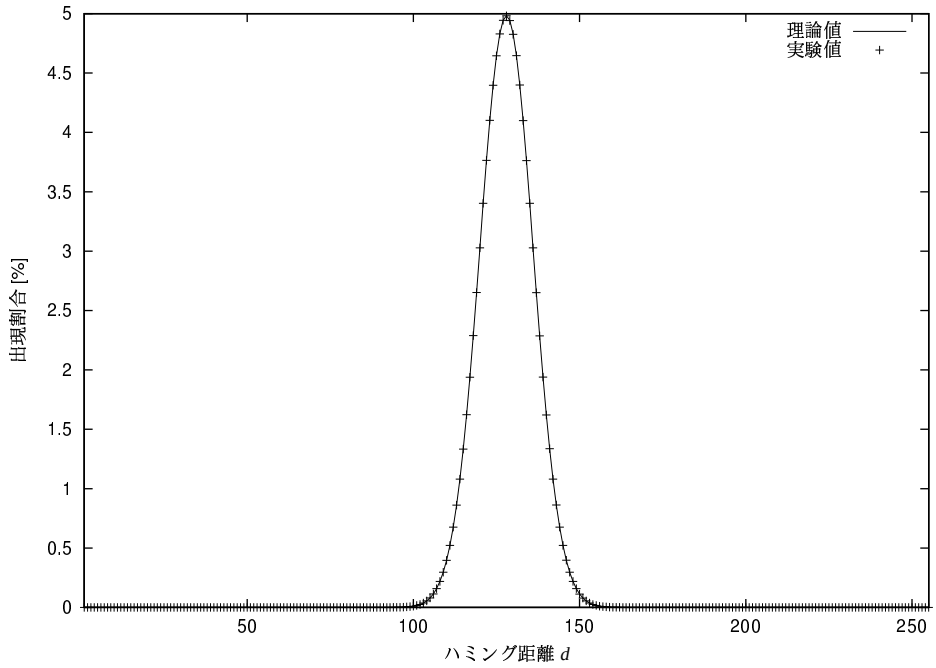

図 2: SHA-256 におけるメッセージの変化がハッシュ値に及ぼす影響 (サンプル数: 2^{26} 対)

2. $h(m)$ が与えれたとき，$h(m)$ の原像 m を発見することが難しい．

後者について，3.6 章で述べる．

3.2 メッセージの変化がハッシュ値に及ぼす影響

任意長のビット系列を n ビットに写像する一様分布のランダムな写像 h を考える．二つのビット系列 m_1, m_2 の写像されたビット系列 $h(m_1), h(m_2)$ のハミング距離が d となる確率 $P_n(d)$ は，

$$P_n(d) = \frac{\binom{n}{d}}{2^n}$$

上記の確率 $P_n(d)$ は，$d = n/2$ とき最大となる．これは，入力のビット系列のハミング距離に関わらず，出力のビット系列の各ビットが変化する確率が $1/2$ であることを意味する．

SHA-256, 512 も上記の性質を満たすべきである．そこで，ハミング距離が 1 の二つのメッセージ m_1, m_2 の SHA-256, 512 によるハッシュ値 $h(m_1), h(m_2)$ のハミング距離を計算機実験で調べた．その結果を図 2, 図 3 に示す．図 2, 図 3 の理論値とは，各々 $P_{256}(d) * 100, P_{512}(d) * 100$ の値である．図 2, 図 3 から，ハッシュ値 (出力のビット系列) の変化は理論値どおりであることが分かる．

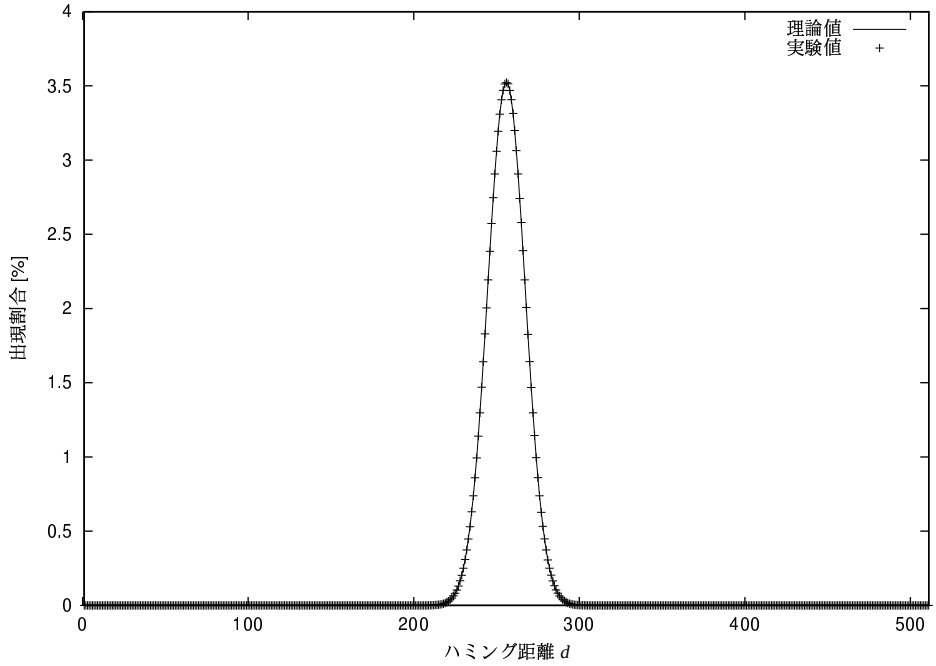

図 3: SHA-512 におけるメッセージの変化がハッシュ値に及ぼす影響 (サンプル数: 2^{23} 対)

3.3 BP 攻撃法による衝突困難性の検証

SHA-256, 512 に対して，BP 攻撃法を行った結果を表 4 示す．表 4 のメッセージ数は，

表 4: SHA-256, 512 への BP 攻撃

	メッセージ数 [個]	衝突したメッセージ数 [個]
SHA-256	2^{25} $(\approx 3.36 \times 10^7)$	0
SHA-512	1.5×2^{23} $(\approx 1.26 \times 10^7)$	0

使用した計算機の RAM に記憶できる最大のメッセージ数である．SHA-256, 512 のいずれの場合も衝突するメッセージを発見できなかった．

SHA-256 が任意長のメッセージを 256bits に写像する理想的なランダム関数であると仮定した場合，この実験で衝突するメッセージが発見できる確率 $p_{256}(2^{25})$ は，以下のように見積もることができる．

$$\begin{aligned} p_{256}(2^{25}) &= 1-(1-\frac{1}{2^{256}})(1-\frac{2}{2^{256}})\cdots(1-\frac{2^{25}-1}{2^{256}}) \\ &< 1-(1-\frac{2^{25}}{2^{256}})^{2^{25}} \\ &\approx 1-(1-\frac{2^{25}\cdot 2^{25}}{2^{256}}) \end{aligned}$$

$$= \frac{1}{2^{206}}$$

同様に，SHA-512 の場合の確率 $p_{512}(2^{23}+2^{22})$ は以下のようになる．

$$\begin{aligned} p_{512}(2^{23}+2^{22}) &= 1-(1-\frac{1}{2^{512}})(1-\frac{2}{2^{512}})\cdots(1-\frac{2^{23}+2^{22}-1}{2^{512}}) \\ &< 1-(1-\frac{2^{24}}{2^{512}})^{2^{24}} \\ &\approx 1-(1-\frac{2^{24}\cdot 2^{24}}{2^{512}}) \\ &= \frac{1}{2^{464}} \end{aligned}$$

この実験ではメッセージ数が非常に少ないので，この実験結果から，SHA-256, 512 が任意長のメッセージを 256bits, 512bits に写像する理想的なランダム関数であるということは言えない．さらに，BP 攻撃法を実行することにより，SHA が任意長のビット系列を固定長のビット系列に写像する理想的なランダム関数であるかどうかを検証することは，現実的でない．なぜなら，BP 攻撃法を実行することが現実的でないように，ハッシュ値のビット長を設定しているからである．

3.4 並列処理による高速化と安全性

SHA-1 の 1 ステップあたりのメッセージ拡張処理も含めた算術・論理演算回数は，高々 14 回である．SHA-1 の圧縮関数は 80 ステップあるので，高々1120 回の算術・論理演算が必要である．一方，SHA-1 のアルゴリズムは並列処理に向いていることが知られている [3]．全て算術・論理演算が同じ時間で実行可能かつ完全に並列動作可能な算術・論理演算器を 7 つ使用できるならば，メッセージ拡張処理も含めて 2 回の演算時間で 1 ステップを終了することができる，つまり，ハッシュ値の計算時間 1/7 にすることができる．ただし，現在のところ，並列動作可能な 7 つの演算器をもつ汎用 CPU はなく，そのような演算器をもつプロセッサを実現することは容易ではないだろう [3].

しかし，もし攻撃者がそのようなプロセッサを作成して SHA-1 に対して BP 攻撃法を行った場合，2^{80} 個のメッセージの生成時間を 1/7 に短縮できる．このように，ハッシュ関数のアルゴリズムの内部処理が並列化できるならば，それが BP 攻撃法に対する安全性の低下の原因となる可能性がある．

SHA-256 のアルゴリズムの並列化の可能性を考えよう．SHA-256 のアルゴリズムにおける 1 ステップで更新される変数は式 (6) と式 (10) の $a[t+1], e[t+1]$ であり，その他の変数は直前のステップの別の変数の代入である．したがって，式 (6) と式 (10) の計算を “いつから開始か?” という点が重要である．

式 (6) の計算には，$e[t]$ の値が必要である．$e[t]$ は $e[t] = d[t-1]+T_1[t-1]$ であるので，直前のステップで更新された値である．したがって，$T_1[t-1]$ の計算が終了しなければ，$e[t]$ を計算することができない．再帰的に，$T_1[t-1]$ の計算には $e[t-1]$ が必要であり，$e[t-1]$ は $T_1[t-2]$ の計算が終了していなければならない．また，式 (10) の計算には，$a[t]$ の値が

必要である．そして，$a[t]$ は直前のステップで更新された値なので，$T_1[t-1], T_2[t-1]$ の計算が終了しなければ，$a[t]$ は求めることができない．したがって，その直前のステップの計算が終了する前に，計算可能な部分としては，

- 式 (4) の $h[t] + K_t^{\{256\}} + w[t]$ の計算
- 式 (5) の $\mathrm{Maj}(a[t], b[t], c[t])$ の一部の計算

だけである．このように，SHA-256 では，その直前のステップの計算が終了しないと，そのステップの大部分の計算を開始することができない．

以上のことより，SHA-256 では，仮に並列動作可能な多くの演算器があったとしても，それらを有効に使うことはアルゴリズム的に難しい．SHA-1 の場合，特別なプロセッサの使用により BP 攻撃法への安全性がやや低下する場合があるが，SHA-256 の場合，そのような安全性の低下はほとんどない．なお，SHA-512 の場合も，変数のビット長が異るだけなので，SHA-256 と同様の結論を得ることができる．

SHA-1 では，あるステップで更新される変数を計算するために使われる変数の値がそのステップより数ステップ前にすでに決定されているため，優れた並列化が可能になっている．一方，SHA-256, 512 では，あるステップで更新される変数を計算するために使われる変数の値が，直前のステップの計算により決定される．この違いが，SHA-1 と SHA-256, 512 のアルゴリズム的な並列可能性の違いの原因である．

ただし，SHA-256, 512 のアルゴリズム中の演算の並列処理が全く不可能ではないことに注意しよう．例えば，上述の演算は，直前のステップと並列に計算することが可能である．また，式 (1) から，2 ステップ前のメッセージが得られていれば，メッセージ拡張処理を開始することができる．現在の汎用 CPU は，数個の演算を並列に処理することができる．よって，SHA-256, 512 のアルゴリズムは，現在の汎用 CPU の並列処理能力を活用する程度の並列処理は可能なアルゴリズムである．また，Pentium III 上の実装で，SHA-1, 256, 512 は，MD5, RIPEMD と比較して，複数のメッセージの並列処理に向いていることも報告されている [9].

したがって，SHA-256, 512 のアルゴリズム的な並列処理可能性が安全性と処理速度に及ぼす影響は，次のようにまとめることができる: SHA-256, 512 のアルゴリズムは，非常に高度な並列処理能力をもつプロセッサがあったとしても，BP 攻撃により安全性が低下するほどの並列化はアルゴリズム的に難しくなるように設計されているが，現在のプロセッサの並列処理能力を活用すると，ハッシュ値の計算速度の向上が期待できるように設計されている．

3.5 既知の攻撃法の適用

この章では，これまでに提案された他のハッシュ関数への代表的な攻撃法を SHA-256, 512 へ適用することを検討する．

Dobbertin の MD4 の衝突探索アルゴリズム Dobbertin は，MD4 で衝突する二つのメッセージ発見する効率的なアルゴリズムを提案している [6]．このアルゴリズムは極め

て強力であり，MD4 が衝突困難な関数でないことを明らかにした．しかし，このアルゴリズムは MD4 の仕様に深く依存しているので，このアルゴリズムを SHA-256, 512 へ適用することは極めて難しい．特に，以下の部分が適用を難しくしている．

- MD4 にはメッセージ拡張処理がないが，SHA-256, 512 にはメッセージ拡張処理がある．
- MD4 では，1 ステップで更新される変数は一つだが，SHA-256, 512 では，それが二つある．
- MD4 では，1 ステップ中のビットの巡回シフトは 1 回だが，SHA-256, 512 では，それが 6 回ある．

このアルゴリズムは，MD5 の擬似衝突を求めるアルゴリズム [6] や短縮された RIPEMD の衝突を求めるアルゴリズム [7] に発展したが，それらを SHA-256, 512 へ適用することは同様の理由で極めて難しい．

Chabaud と Joux の SHA-0 の差分を利用した衝突探索アルゴリズム Chabaud と Joux は，差分攻撃法を利用して SHA-0 の衝突探索アルゴリズムの計算量を 2^{61} と理論的に評価した [4]．SHA-0 のハッシュ値は 160bits であるから，このアルゴリズムは BP 攻撃法よりも効率の良い攻撃法である．しかし，このアルゴリズムを SHA-1 に適用しても BP 攻撃法よりも効率良く衝突を発見することはできない．その原因は，SHA-1 のメッセージ拡張処理における 1 ビットの巡回シフトにある．つまり，メッセージ拡張処理における 1 ビットの巡回シフトによって，SHA-1 の差分攻撃法に対する安全性は SHA-0 よりも向上している．

Chabaud と Joux のアルゴリズムを SHA-256, 512 へ適用することは極めて難しい．ハッシュ値の長さの違いだけでなく，SHA-256, 512 のメッセージ拡張処理は，SHA-1 のそれよりもさらに複雑になっているためである．

ラウンド数を削減した SHA-1 の衝突探索アルゴリズム 1 ラウンド (ステップ 1 からステップ 20) のみの SHA-1 の衝突探索アルゴリズムが提案されている [8]．このアルゴリズムを SHA-256, 512 に適用することは，以下の理由により極めて困難である．

- SHA-256, 512 のメッセージ拡張処理が SHA-1 のそれよりも多くの巡回シフトを含んでいる．
- MD4 では，1 ステップで更新される変数は一つだが，SHA-256, 512 では，それが二つある．さらに，その二つの変数が直前のステップで更新された値に依存している．

いずれの攻撃法に対しても，メッセージ拡張処理が安全性の向上に寄与していることが分かる．

3.6 原像探索困難性について

SHA-256, 512 の場合，メッセージ m とそのハッシュ値 $h(m)$ が与えられたとき，$h(m) = h(m')$ となるメッセージ m' が存在する．そのような m' を求めることが計算量的に困難な場合，そのハッシュ関数は一方向性であるという．この章では，SHA-256 のハッシュ値 $h(m)$ のみが与えられたとき，原像である m を求めることを検討する．最初に以下の補題を証明する．

補題 1 SHA-256 において，式 (6) ∼ 式 (13) の $a[t], b[t], \ldots, h[t]$ と $a[t+4], b[t+4], \ldots, h[t+4]$ が与えられたとき，$W[t], W[t+1], W[t+2], W[t+3]$ の値は一意であり，効率良く求められる．

証明 1 SHA-256 のアルゴリズムから，$h[t+4] = d[t] + T1[0]$ であるので，$T1[0]$ の値を求めることができる．そのとき，$W[t]$ の値は，以下の式で計算できる．

$$W[t] = T1[t] - (h[t] + \Sigma_1^{256}(e[t]) + \mathrm{Ch}(e[t], f[t], g[t]) + K_t^{\{256\}})$$

式 (5) ∼ 式 (13) を用いて，$a[t+1], b[t+1], \ldots, h[t+1]$ を計算する．

次に，$T1[1] = g[t+4] - c[t]+$ から $T1[1]$ を計算する．そのとき，$W[t+1]$ の値は，以下の式で計算できる．

$$W[t+1] = T1[t+1] - (h[t+1] + \Sigma_1^{256}(e[t+1]) + \mathrm{Ch}(e[t+1], f[t+1], g[t+1]) + K_{t+1}^{\{256\}})$$

式 (5) ∼ 式 (13) を用いて，$a[t+2], b[t+2], \ldots, h[t+2]$ を計算する．同様の計算を繰り返すことにより，$W[t+2], W[t+3]$ を計算することができる．

上記の補題より，以下の定理を直ちに得ることができる．

定理 1 SHA-256 において，式 (6) ∼ 式 (13) の $(a[t], b[t], \ldots, h[t])$, $(a[t+4], b[t+4], \ldots, h[t+4])$, … $(a[t+16], b[t+16], \ldots, h[t+16])$ が与えられたとき，$W[t], W[t+1], \ldots, W[t+16]$ の値は一意であり，効率良く求められる．

上記の定理において，もし $t = 0$ ならば，求めた $W[0], W[1], \ldots, W[16]$ はメッセージそのものである．一般には，$W[t], W[t+1], \ldots, W[t+16]$ から式 (1) を用いて，$W[0], W[1], \ldots, W[16]$ を計算する必要がある．しかし，16 個の式 (1) の連立方程式には，シフト，巡回シフト，論理演算，法加算の演算が混在しているため，それを解く効率の良いアルゴリズムは自明ではなく，今回の評価では，そのアルゴリズムを考案することができなかった．なお，上記と同様の議論が SHA-512 に対しても成立する．

次に，SHA-1 に対して，同様の攻撃を考える．同様の記法を用いると，$(a[t], b[t], \ldots, e[t])$, $(a[t+5], b[t+5], \ldots, e[t+5])$, … $(a[t+15], b[t+15], \ldots, e[t+15])$ が与えられたとき，$O(2^{32})$ の計算量で，$W[t], W[t+1], \ldots, W[t+16]$ の値は一意に決定することができる．さらに，SHA-1 のメッセージ拡張処理は巡回シフトと排他的論理和だけからなり，この二つの演算は可換なので，$t \neq 0$ のでも $W[0], W[1], \ldots, W[16]$ を容易に決定することができる．

アルゴリズムの実行中の変数の値が攻撃者に知られることは，現実には起りにくい．しかし，仮にそのようなことが起ったとしても，SHA-256, 512 の場合，そのハッシュ値の原像を求めることは容易ではない．この章で考察したような攻撃法に対しても，SHA-256, 512 のメッセージ拡張処理が安全性の向上に役立っていることがわかる．

3.7 Kelsey 法の評価

2.3.2 章で述べた Kelsey 法の評価を述べる．Draft FIPS-180-2 には，ハッシュ関数を鍵付ハッシュのメッセージ認証に利用することを述べているので，Kelsey のコメントは検討する価値がある．[12] では，Kelsey 法は安全性と処理速度を損なうことなく，extension property をなくすことができると述べられている．処理速度が著しく低下しないことは自明であるが，安全性については，証明はされていない．

一方，安全なハッシュ関数を用いて，効率的かつ安全性が証明可能な鍵付ハッシュ関数の構成法が知られている [1]．この構成法を Kelsey 法を比較した場合，処理速度は Kelsey 法の方が優れているが，安全性の面で Kelsey 法はまだ検討が十分ではない．したがって，Kelsey 法の安全性の証明がされるまでは，[1] の構成法を利用する方がよいであろう．

4 結論

この報告では，SHA-256, 384, 512 の安全性の検討を行った．現在のところ，SHA-256, 384, 512 の致命的な欠点はもちろん，安全性を疑わせるような点も見当たらない．しかし，SHA-256, 384, 512 のアルゴリズムは提案されてから日が浅く，安全性が十分に研究されているとは言えない．今後も引き続き安全性を検討する必要がある．

SHA-256, 384, 512 の設計は，SHA-1 の設計とはかなり異っている点に十分に注意をする必要がある．その設計の変更は，いくつかの攻撃に対して SHA-256, 384, 512 の安全性の向上に役立っていることが今回の検討で確認できた．特に，メッセージ拡張処理が複雑になっている点が既存の攻撃法に対する安全性の向上に大きく寄与している．しかし，逆に，設計の変更点が安全性の低下の要因になる可能性も否定はできない．Jonsson のコメントにもあるように，NIST が SHA-256, 384, 512 の安全性評価を公開し，SHA-1 と異る設計にした理由を明らかにすることが望まれる．

参考文献

[1] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," Advances in Cryptology - CRYPTO'96, Lecture Notes in Computer Science, vol. 1109, pp. 1–15, 1996.

[2] M. Bellare and P. Rogaway, "Collision-resistant hashing: Towards making UOWHFs practical," Advances in Cryptology - CRYPTO'97, Lecture Notes in Computer Science, vol. 1294, pp. 470–484, 1997.

[3] A. Bosselaer, R. Govaerts, and J. Vandewalle, "SHA: A design for parallel architectures ?," Advances in Cryptology - EUROCRYPT'97, Lecture Notes in Computer Science, vol. 1233, pp. 348–362, 1997.

[4] F. Chabaud and A. Joux, "Differential collisions in SHA-0," Lecture Notes in Computer Science Advances in Cryptology - CRYPTO'98, vol. 1462, pp. 56–71, 1998.

[5] I. B. Damgård, "A design principle for hash functions," Advances in Cryptology - CRYPTO'89, Lecture Notes in Computer Science, vol. 435, pp. 416–427, 1990.

[6] H. Dobbertin, "Cryptanalysis of MD4," Lecture Notes in Computer Science Fast Software Encryption, vol. 1039, pp. 53–69, 1996.

[7] H. Dobbertin, "RIPEMD with two-round compress function is not collision-free," Journal of Cryptology, vol. 10, no. 1, pp. 51–69, 1997.

[8] 木村太一，桑門秀典, 田中初一, "SHA の高速衝突探索の一手法," 第 22 回情報理論とその応用シンポジウム予稿集, pp. 463-466, 1999.

[9] J. Nakajima and M. Matsui, "Parallel implementations of dedicated hash functions," Proceedings of the 2002 Symposium on Cryptography and Information Security, vol. II, pp. 867–872, 2002.

[10] National Institute of Standards and Technology, "Secure hash standard," NIST FIPS PUB 180-1, 1993.

[11] National Institute of Standards and Technology, "Secure hash standard," Federal Information Processing Standards Publication 180-2, http://csrc.nist.gov/encryption/shs/dfips-180-2.pdf, 2001.

[12] National Institute of Standards and Technology, "Public comments on the draft federal information processing standard (FIPS) draft FIPS 180-2, secure hash standard (SHS)," http://csrc.nist.gov/encryption/shs/dfips-180-2-comments1.pdf, 2001.

[13] V. Shoup, "A composition theorem for universal one-way hash functions," Advances in Cryptology - EUROCRYPTO 2000, Lecture Notes in Computer Science, vol. 1807, pp. 445–452, 2000.