IPA®
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

共通鍵暗号をベースとした
ハッシュ関数安全性評価手法の調査
調査報告書

2008 年 5 月

独立行政法人　情報処理推進機構

目 次

1 ハッシュ関数の安全性 **3**
1.1 基本安全性要件 . . . 3
1.2 ハッシュ関数族の安全性要件 . . . 3

2 証明可能なハッシュ関数の安全性評価 **4**
2.1 調査目的 . . . 4
2.2 The Models and the definitions for Hash Functions . . . 4
2.2.1 The Standard Model . . . 5
2.2.2 The Random Oracle Model . . . 5
2.2.3 The Ideal Cipher Model . . . 6
2.3 Block-cipher based hash function with provable security . . . 6
2.4 ハッシュ関数の定義域拡大 . . . 8
2.4.1 定義域拡大の安全性証明 . . . 10
2.5 まとめ . . . 13

3 実装性能重視型ハッシュ関数の安全性評価 **13**
3.1 調査目的 . . . 13
3.2 定義域拡大＋圧縮関数型ハッシュ関数の安全性評価 . . . 13
3.2.1 Tiger . . . 15
3.2.2 FORK . . . 17
3.2.3 Tiger と FORK 以外のハッシュ関数 . . . 21
3.3 PANAMA 型ハッシュ関数の安全性評価 . . . 24
3.3.1 PANAMA に対する攻撃 . . . 24
3.3.2 Grindahl に対する衝突攻撃 . . . 25
3.4 まとめ . . . 28

4 共通鍵暗号をベースとしたハッシュ関数の安全性の評価ツール仕様の検討 **28**
4.1 調査目的 . . . 28
4.2 検討対象 . . . 29
4.3 検討項目 . . . 29
4.3.1 コンセプト . . . 29
4.3.2 課題抽出 . . . 29
4.3.3 機能要件 . . . 30
4.3.4 モジュール構成 . . . 30
4.4 仕様検討 . . . 31
4.5 まとめ . . . 32

5 ハッシュ関数 MAME の安全性評価 **32**
5.1 調査目的 . . . 32
5.2 MAME の仕様 . . . 32
5.2.1 暗号化関数 . . . 32

5.2.2 ハッシュ関数の構成 35
5.3 MAME の圧縮関数モード部の安全性評価 36
5.4 MAME のブロック暗号部の安全性評価 37
5.4.1 AES 選定プロセスにおける finilist の安全性評価 37
5.4.2 AES 会議開催中に提案された解読法 37
5.4.3 AES 会議開催後に提案された解読法 39
5.5 まとめ 39

1　ハッシュ関数の安全性

ハッシュ関数は、任意長のメッセージを固定長の出力に写す関数である。本章では、ハッシュ関数が満たすべき基本安全性要件を述べるとともに、近年新しく議論されている、ハッシュ関数を族として考えた際の安全性要件も述べる。

1.1　基本安全性要件

ハッシュ関数が満すべき基本安全性要件は、以下である。

- 衝突耐性
  攻撃者が、$h(M) = h(M')$ なる (M, M') のメッセージ対を見付けることが難しい。
- 第二原像耐性
  メッセージ M を与えられた攻撃者が、$h(M) = h(M')$ なる M とは異なるメッセージ M' を見付けることが難しい。
- 原像耐性
  メッセージ M に対し、$Y = h(M)$ とし、Y を与えられた攻撃者が $Y = h(M')$ なるメッセージ M' を見付けることが難しい。

1.2　ハッシュ関数族の安全性要件

ハッシュ関数族には、複数の安全性概念が存在する。ここでは、Rogaway 等 [83] により検討されている安全性概念を導入する。以下で記述する概念は、ハッシュ関数族 $H : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ を対象としたものである。ここで、鍵空間 $\mathcal{K}$, ターゲット空間 $\mathcal{Y}$ は、ビット列の有限集合である。メッセージ空間 $\mathcal{M}$ は無限集合でもよいが、少なくともある λ に対して $\{0,1\}^\lambda \subset \mathcal{M}$ とする。

- 衝突耐性 (Coll)
  一様確率で選ばれた $K \in \mathcal{K}$ に対して、K を与えられた攻撃者が、$H(K, M) = H(K, M')$ なる (M, M') のメッセージ対を見付けることが難しい。
- 第二原像耐性 [λ] (Sec[λ])
  一様確率で選ばれた $K \in \mathcal{K}$ 及び $M \in \{0,1\}^\lambda$ に対して、K 及び M を与えられた攻撃者が、$H(K, M) = H(K, M')$ なるメッセージ M' を見付けることが難しい。
- 原像耐性 [λ] (Pre[λ])
  一様確率で選ばれた $K \in \mathcal{K}$ 及び $M \in \{0,1\}^\lambda$ に対して、$Y = H(K, M)$ とし、K 及び Y を与えられた攻撃者が $Y = H(K, M')$ なるメッセージ M' を見付けることが難しい。
- everywhere-第二原像耐性 (eSec)
  攻撃者がメッセージ $M \in \mathcal{M}$ を事前に選び、一様確率で選ばれた $K \in \mathcal{K}$ に対して、K を与えられた攻撃者が $H(K, M) = H(K, M')$ なるメッセージ M' を見付けることが難しい。

- everywhere-原像耐性 (ePre)
  攻撃者が原像 $Y \in \mathcal{Y}$ を事前に選び、一様確率で選ばれた $K \in \mathcal{K}$ に対して、K を与えられた攻撃者が $Y = H(K, M')$ なるメッセージ M' を見付けることが難しい。

- always-第二原像耐性 [λ] (aSec[λ])
  攻撃者が鍵 $K \in \mathcal{K}$ を事前に選び、一様確率で選ばれた $M \in \{0,1\}^\lambda$ に対して、M を与えられた攻撃者が $H(K, M) = H(K, M')$ なるメッセージ M' を見付けることが難しい。

- always-原像耐性 [λ] (aPre[λ])
  攻撃者が鍵 $K \in \mathcal{K}$ を事前に選び、一様確率で選ばれた $M \in \{0,1\}^\lambda$ に対して、$Y = H(K, M)$ とし、Y を与えられた攻撃者が $Y = H(K, M')$ なるメッセージ M' を見付けることが難しい。

上記のうち基本的な 3 つの性質は原像耐性、第二原像耐性、衝突耐性である。このうち、原像耐性及び第二原像耐性に対しては、everywhere と always という 2 つずつの変形版がある。everywhere-原像耐性 (everywhere-第二原像耐性) は、攻撃者が原像 (メッセージ) を選択できるようになっている。Naor 等によるユニバーサル・ハッシュ関数及び Bellare 等によるターゲット衝突耐性は、eSec と同等であることが知られている。一方、always-原像耐性 (always-第二原像耐性) は、攻撃者が鍵を選択できるようになっている。このため、ハッシュ関数族に属する全てのハッシュ関数が、鍵に依らず、Pre (Sec) となっていることが必要となる。SHA1, SHA256 等のハッシュ関数は、陽には鍵を持っていないため、これらのハッシュ関数を用いた安全性の議論では always-原像耐性 (always-第二原像耐性) が重要になる。

上記に挙げた概念以外にも、多重衝突、擬似ランダムオラクル性等があるが、本節では説明を省略する。

2　証明可能なハッシュ関数の安全性評価

2.1　調査目的

本章では、近年提案された、ある構成要素がその下で用いる要素の安全性に対するなんらかの仮定の下で証明可能安全であるということを特徴にもつようなハッシュ関数を取り上げ、安全性根拠の調査および安全性要件の検討を行い、安全性評価手法の調査・検討を行う。

2.2　The Models and the definitions for Hash Functions

In this section, we describe definitions and the models for hash functions that are provably secure, follwing and summarising the result by [9].

Definition 1 (Block ciphers) *Let $\kappa, n \geq 1$ be numbers. A blockcipher is a map $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ where, for each $k \in \{0,1\}$, the function $E_k = E$ is a permutation on $\{0,1\}^n$. Parameter n is called the blocksize of E. If E is a blockcipher then E^{-1} is its inverse, where $E_k^{-1}(y)$ is the string x such that $E_k(x) = y$. Let Bloc(κ, n) be the set of all block ciphers $E : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$. Choosing a random element of Bloc(κ, n) means that for each $k \in \{0,1\}^\kappa\}$ one chooses a random permutation $E_k(\cdot)$.*

Definition 2 (Block cipher based hash functions) *A block cipher based hash functions is a map H: Bloc$(\kappa, n) \times D \to R$ where $\kappa, n, c \geq 1, D \subseteq \{0,1\}^*$, and $R = \{0,1\}^c$. The function H must be given by a program that, given M, computes $H^E(M) = H(E, M)$ using an E-oracle. Hash function*

f:Bloc$(\kappa, n) \times D \to R$ is a compression function if $D = \{0,1\}^a \times \{0,1\}^b$ for some $a, b \geq 1$ where $a+b \geq c$. Fix $h_0 \in \{0,1\}^a$. The iterated hash of compression function f: Bloc$(\kappa, n) \times (\{0,1\}^a \times \{0,1\}^b) \to \{0,1\}^a$ is the hash function H: Bloc$(\kappa, n) \times (\{0,1\}^b)^* \to \{0,1\}^a$ defined by $H^E(m_1 \cdots m_l) = h_l$ where $h_i = f^E(h_{i-1}, m_i)$. Set $H^E(\epsilon) = h_0$. We often omit the superscript E to f and H.

We write $x \xleftarrow{r} S$ for the experiment of choosing a random element from the finite set S and calling it x. An adversary is an algorithm with access to one or more oracles. To quantify the collision resistance of a block cipher based hash function H, we instantiate the block cipher by a randomly chosen $E \in \mathrm{Bloc}(\kappa, n)$. An adversary A is given oracles for $E(\cdot,\cdot)$ and $E^{-1}(\cdot,\cdot)$ and wants to find a collision for H^E - that is, $M \neq M'$ where $M \neq M'$ but $\wedge H^E(M) = H^E(M')$. We look at the number of queries that the adversary makes and compare this with the probability of finding a collision.

Definition 3 (Collision Resistance) *Let H be a block cipher based hash function, H: Bloc$(\kappa, n) \times D \to R$, and let A be an adversary. Then the advantage of A in finding collisions in H is the real number*

$\mathbf{Adv}_H^{\mathrm{coll}}(A) = Pr[E \xleftarrow{r} \mathrm{Bloc}(\kappa, n);\ (M, M') \xleftarrow{r} A^{E, E^{-1}}:\ M \neq M' \wedge H^E(M) = H^E(M')]$

For $q \geq 1$ we write

$$\mathbf{Adv}_H^{\mathrm{coll}}(q) = \max_{\mathcal{A}} \left\{ \mathbf{Adv}_H^{\mathrm{coll}}(\mathcal{A}) \right\} \ ,$$

where the maximum is taken over all adversaries that ask at most q oracle queries to (E-queries + E^{-1}-queries).

Before we can prove the security of a cryptographic system or object, we must specify what model we are using.

2.2.1 The Standard Model

The most common model used in modern cryptography is the so-called "standard model". We abstract our communications system typically as a reliable but insecure channel. We have not been able to achieve most common cryptographic goals in the standard model without making additional complexity-theoretic hardness assumptions. The common assumptions are typically that factoring the product of large primes is hard, or that AES is a good pseudo-random permutation (PRP) [55]. The standard model is usually well-accepted in our community despite the fact that proofs done in this model rest upon unproven assumptions.

2.2.2 The Random Oracle Model

Let $\boldsymbol{F}_{n',n} = \{f \mid f : \{0,1\}^{n'} \to \{0,1\}^n\}$. In the random oracle model, the function f is assumed to be randomly selected from $\boldsymbol{F}_{n',n}$. The computation of f is simulated by the following oracle.

The oracle f first receives an input x_i as a query. Then, it returns a randomly selected output y_i if the query has never been asked before. It keeps a table of pairs of queries and replies, and it returns the same reply to the same query.

When proofs in the standard model are provably impossible (eg, see [70]), we often resort to proofs using an alternative model. By far the best-known is the "Random-Oracle Model." The random-oracle model was used for some time before being formalized by Bellare and Rogaway [4], and continues to see

widespread use today ([23, 63]). In the random-oracle model we have a public random function, accessible to all parties, which typically accepts any string from $\{0,1\}^*$ and outputs n bits. For each element in its domain, the corresponding n-bit output is uniform and independent from all other outputs.

Of course random oracles do not exist in practice, and if the schemes proven secure in the random-oracle model are going to be put into use, we must choose some object to implement the random oracle. This step is called "instantiation." Most often, random oracles are instantiated with cryptographic hash functions such as SHA-1[71].

2.2.3 The Ideal Cipher Model

Blockciphers are a common building block for cryptographic protocols. In the standard model the associated assumption for blockciphers is that they are "pseudo-random permutations" (PRPs). By this we mean (informally) that an n-bit blockcipher under a secret randomly-chosen key is computationally indistinguishable from a randomly-chosen n-bit permutation. Proofs conducted using this assumption typically give reductions showing that if an adversary breaks some scheme, then there exists an associated adversary that can efficiently distinguish the underlying blockcipher from random.

In certain cases it can be shown that blockcipher-based schemes we believe to be secure cannot have a proof of security using only the PRP assumption in the standard model [93]. In this case we are faced with either abandoning attempts at a proof, or using an alternate model. The blockcipher analog for the random-oracle model is variously called the "Black-Box Model," or the "Ideal-Cipher Model." We will prefer the latter name in this report. Though not as widely-used as the random-oracle model, the ideal-cipher model dates back to Shannon [91] and has been used in a variety of settings (see, for example [95, 62, 22, 43, 19, 11, 40]). In the ideal-cipher model we think of a blockcipher E with k-bit key and n-bit blocksize as being chosen uniformly from the set of all possible blockciphers of this form. For each key, there are $2n!$ permutations, and since any permutation may be assigned to a given key, there are $(2^n!)^{2^k}$ possible blockciphers. When we instantiate our black box, it becomes some particular blockcipher. AES with a 128-bit key is one choice from the $(2^{128}!)^{2^{128}}$ blockciphers we could have chosen.

The ideal-cipher model has been used in a variety of settings, and like the random-oracle model, some researchers question the wisdom of its use. A common argument against the ideal-cipher model is that most real-world blockciphers have distinguishing patterns which would exist with exceedingly small probability in a collection of random permutations. The key complementation property of DES is a typical example of this [55]. Although no such properties are currently known for AES, some blockcipher experts who are comfortable with the assumption that AES is a good PRP are reluctant to model AES as ideal because of practical concerns: the AES key schedule, for instance, is quite simple and it perhaps contains related-key properties we have not yet discovered.

2.3 Block-cipher based hash function with provable security

Definition 4 *A hash function $H : \{0,1\}^* \to \{0,1\}^\ell$ usually consists of a compression function $F : \{0,1\}^\ell \times \{0,1\}^{\ell'} \to \{0,1\}^\ell$ and a fixed initial value $h_0 \in \{0,1\}^\ell$. An input m is divided into the ℓ'-bit blocks $m_1, m_2, \ldots, m_l$. Then, $h_i = F(h_{i-1}, m_i)$ is computed successively for $1 \le i \le l$ and $h_l = H(m)$. H is called an iterated hash function.*

表 1: Provable secure hash functions

Name	Authors	Rate
DHF	Hirose	1/4
MDC-2	Brachtl *et al*	1/2
MDC-4	Brachtl *et al*	1/4
-	Knudsen and Preneel	1/24
-	Merkle	0.276
tandem/abreast Davies-Meyer	Lai and Massey	1/2
-	Nandi *et al*	2/3
-	Gauravaram, Millan and May	1

Definition 5 *An iterated hash function is called a double-block-length (DBL) hash function if its output length is twice larger than the block length.*

Definition 6 *Let F be a compression function composed of a block cipher. For an iterated hash function composed of F, the rate r defined below is often used as a measure of efficiency:*

$$r = \frac{|m_i|}{\textit{(the number of block-cipher calls in } F\textit{)} \times n} \ .$$

Definition 7 *A hash/compression function is optimally collision-resistant if any attack to find its collision is at most as efficient as the birthday attack.*

We review the previous work on hash functions composed of block ciphers in the following.

Knudsen and Preneel studied the schemes to construct secure compression functions based on error-correcting codes [45, 46, 47]. It is an open question whether their schemes are optimally collision-resistant or not. Knudsen *et al* [48] discussed the insecurity of DBL hash functions with the rate 1 composed of (n, n) block ciphers. Hohl *et al* [36] discussed the security of compression functions of DBL hash functions with the rate 1/2. Merkle [62] presented three DBL hash functions composed of DES with the rates at most 0.276. They are optimally collision-resistant in the ideal cipher model. MDC-2 and MDC-4 [12] are also DBL hash functions composed of DES with the rates 1/2 and 1/4, respectively. Lai and Massey proposed the tandem/abreast Davies-Meyer [51] based on an $(n, 2n)$ block cipher and their rates are 1/2. It is an open question whether the four schemes are optimally collision-resistant or not. Hirose [32] presented a large class of DBL hash functions with the rate 1/2, based on $(n, 2n)$ block ciphers. They were shown to be optimally collision-resistant in the ideal cipher model. However, his construction requires two independent block ciphers, which makes the results less attractive. Nandi *et al* [69] also proposed an interesting construction with the rate 2/3. However, they are not optimally collision-resistant. Futhermore, Knudsen and Muller [44] presented some attacks against it. Gauravaram *et al* proposed a new technique to design rate-1 hash functions that can be instantiated with any secure 128-bit block cipher reduced to half the number of rounds [27].

Hirose [33] presented a compression function specified in the following definition.

Definition 8 *Let* $F : \{0,1\}^{2n} \times \{0,1\}^b \rightarrow \{0,1\}^{2n}$ *be a compression function such that* $(g_i, h_i) = F(g_{i-1}, h_{i-1}, m_i)$*, where* $g_i, h_i \in \{0,1\}^n$ *and* $m_i \in \{0,1\}^b$*. F consists of an* $(n, n+b)$ *block cipher E as follows:*

$$
\begin{aligned}
g_i &= F_{\mathrm{U}}(g_{i-1}, h_{i-1}, m_i) && (1)\\
&= e(h_{i-1} \parallel m_i, g_{i-1}) \oplus g_{i-1} && (2)\\
h_i &= F_{\mathrm{L}}(g_{i-1}, h_{i-1}, m_i) && (3)\\
&= E(h_{i-1} \parallel m_i, g_{i-1} \oplus c) \oplus g_{i-1} \oplus c \ , && (4)\\
& && (5)
\end{aligned}
$$

where $\parallel$ *represents concatenation and* $c \in \{0,1\}^n - \{0^n\}$ *is a constant.*

F requires two invocations of E to produce an output. However, these two invocations need only one key scheduling of E. If F is implemented using the AES with 192-bit key-length, then $n = 128$, $b = 64$ and the rate is $1/4$. If implemented using the AES with 256-bit key-length, then $n = b = 128$ and the rate is $1/2$. The following theorem shows the collision resistance of a hash function composed of F in Definition 8.

Theorem 1 *Let* H *be a hash function composed of the compression function* F *specified in Definition 8 and an initial value* (g_0, h_0)*. Then, for every* $1 \leq q \leq 2^{n-2}$*,*

$$
\mathbf{Adv}_H^{\mathrm{coll}}(q) \leq \frac{3\,q^2 + q}{2^{2(n-1)}} \leq \left(\frac{q}{2^{n-2}}\right)^2 \ .
$$

2.4 ハッシュ関数の定義域拡大

Guido Bertoni, Joan Daemen, Michael Peeters and Gilles Van Assche, "Sponge functions," Ecrypt Hash Workshop 2007 [6]

良いハッシュ関数はランダムオラクルのように振舞うべきであり、ランダムオラクルがもたないような脆弱性はもつべきではない。本論文では、衝突の存在という性質に限り、ランダムオラクルと識別可能であるような構成法を提案し、これをスポンジと呼んでいる。そして、いくつかの攻撃について成功確率を計算することにより、ランダムスポンジの強度を評価している。そして、ハッシュ関数や MAC 関数ストリーム暗号に対する安全性要件を記述するためのリファレンスとして、ランダムスポンジを用いることを提案している。

Orr Dunkelman and Bart Preneel,"Generalizing the herding attack to concatenated hashing schemes," Ecrypt Hash Workshop 2007 [21]

連結型ハッシュ関数に対する集め攻撃 (Herding attack) を拡張している。本論文の結果は、より大きなハッシュ関数の集合に適用できる。圧縮関数が二つあるいはそれ以上の個数のデータパスとして記述でき、ただし、最初のデータパスが二番目のデータパスに影響されないような場合に、一般化された herding attack を適用できることを示している。この結果が示唆しているのは、ハッシュ関数の耐性を改良することを目的とするスキームは、さまざまなデータパス間で攪拌を行わなければならないことである。

Elena Andreeva, Gregory Neven, Bart Preneel and Thomas Shrimpton,"Three-property preserving iterations of keyless compression functions," Ecrypt Hash Workshop 2007 [2]

ハッシュ関数の多くは、有限領域の圧縮関数の Merkle-Damgaard 繰り返しに基づいている。最近提案された構成方法 ROX は、会議 FSE2004 で Rogaway らにより示された全七個の安全性概念を証明可能な方法で保存する。しかし、鍵として知られる公開パラメータにより、インデックス化された圧縮関数をもつようなハッシュ関数のファミリに対しては、それは成り立つが、現実的なハッシュ関数は、このようなパラメータをもたない。したがって、これらのスキームを具現化する方法は明らかではない。本論文では、この状況を解決するために Rogaway の人間無知 (Human-ignorance) のアプローチを用い、4 つの異なる繰り返し構造 (それらの二つは連鎖ベースで残りの二つは木構造ベース) を提案している。これらが、基本安全性の 3 つを証明可能な方法で保存することを示している。

Ilya Mironov and Arvind Narayanan,"Domain extensions for random oracles: beyond the birthday-paradox bound," Ecrypt Hash Workshop 2007 [64]

本論文では、圧縮関数をランダムオラクルとして、モデル化したときに、証明可能安全に衝突耐性をもつ新しいスキームを提案している。この構成法のレイトは MDC-2 と同等だが、安全性は、MDC-2 に対して知られている最良の限界より強い。さらに、Maurer らによる枠組みにおいて、ランダムオラクルと識別不可能である構成法であることを示している。

Thomas Shrimpton and Martijn Stam,"Efficient collision-resistant hashing from fixed-length random oracles," Conference Hash Functions [92]

本論文では、圧縮関数をランダムオラクルとして、モデル化したときに、証明可能安全に衝突耐性をもち、一回の繰り返しにつき、一回より多くの理想的名プリミティブの呼び出しを行う 2n ビットから n ビットへ出力する効率的な圧縮関数が可能であることの証拠を提供している。3 つの異なる長さを保つランダムオラクルを用いて、各メッセージブロックにつき、それらを一回ずつ呼び出すような構成法を提案している。

Thomas Ristenpart and Thomas Shrimpton,"Building application-agile hash functions: the MCM construction," Ecrypt Hash Workshop 2007 [80]

証明可能安全な衝突耐性をもち、かつ、ランダムオラクルの振舞のある程度の保証により、より良い安全性が提供できる可能性がある。本論文では、これらのゴールを達成するハッシュ関数を application agile なハッシュ関数と呼んでいる。既存の証明可能なハッシュ関数は一般的にはこれらのゴールの両方を達成してはいない。本論文では、スタンダードモデルの下で証明可能な衝突耐性をもち、同時に、理想的暗号モデルにおいてランダムオラクルと識別不可能なハッシュ関数を与える最初の構成法 (MCM 構成法) を提案している。

2.4.1 定義域拡大の安全性証明

Elena Andreeva, Gregory Neven, Bart Preneel and Thomas Shrimpton, "Seven-Property-Preserving Iterated Hashing: ROX," ASIACRYPT 2007 [3]

任意長のメッセージから固定長のハッシュ値への写像であるハッシュ関数は、多くの場合、固定長メッセージから固定長出力への写像である圧縮関数を繰り返し用いることで構成されている。構成法として、Strengthened Merkle-Damgard 構成法を始めとする表 2 の 11 種類がよく知られている。ハッシュ関数および圧縮関数の安全性概念としては、原像耐性 (Pre)、第二原像耐性 (Sec)、衝突耐性 (Col) が良く知られているが、前者 2 つには everywhere, always という変種 ePre, aPre, eSec, aSec がある。本論文では、前記各構成法が 7 つの各安全性を保存するかどうかを調べている。ある安全性を持つ圧縮関数から構成したハッシュ関数が同じ安全性を持つ場合、構成法はその安全性を保存するという。本論文により、上記 11 種の構成法の中には、7 つの安全性全てを保存するものはないことが分かった。

表 2: 定義域拡大の構成法

構成法	提案者
Strengthened MD	Merkle, Damgøard
Linear	Bellare, Rogaway
XOR-Linear	Bellare, Rogaway
Shoup's	Shoup
Prefix-free MD	Coron, Dodis, Malinaud, Puniya
Randomized	Halevi, Krawczyk
HAIFA	Biham, Dunkelman
Enveloped MD	Bellare, Ristenpart
Strengthened Merkle Tree	Merkle
Tree Hash	Bellare, Rogaway
XOR Tree	Bellare, Rogaway

本論文は、全ての安全性を保つ新しい構成法として、ROX を提案している。ROX はランダムオラクルを用いているが、入力の長さは固定長で短く、呼び出し回数もメッセージブロック数の対数に比例しており、少ないといえる。ROX を含めた各構成法の安全性保存・非保存は、表 3 となっている。

French Saphir Project (Cryptolog, France Telecom, Ecole Normale Superieure, DCSSI and Gemalto),"Revisiting security relations between signature schemes and their inner hash functions," Ecrypt Hash Workshop 2007 [26]

本論文では、ハッシュ関数に対する攻撃が署名スキームの安全性に如何に影響するかに関して、初期検討結果を報告しており、確率的な Hash-and-sign の概念を提案している。さらに様々な関連カテゴリーに署名スキームを分類している。そして Merkle-Damgaard 方式の繰り返し型ハッシュ関数を使うことが決定的 (または、確率的) な署名スキームの安全性に如何に影響するかを決定している。

表 3: 定義域拡大の構成法の安全性

構成法	Coll	Sec	aSec	eSec	Pre	aPre	ePre
Strengthened MD	Y	N	N	N	N	N	Y
Linear	N	N	N	N	N	N	Y
XOR-Linear	Y	N	N	Y	N	N	Y
Shoup's	Y	N	N	Y	N	N	Y
Prefix-free MD	N	N	N	N	N	N	Y
Randomized	Y	N	N	N	N	N	Y
HAIFA	Y	N	N	N	N	N	Y
Enveloped MD	Y	N	N	N	N	N	Y
Strengthened Merkle Tree	Y	N	N	N	N	N	Y
Tree Hash	N	N	N	N	N	N	Y
XOR Tree	?	?	N	?	Y	N	Y
ROX	Y	Y	Y	Y	Y	Y	Y

Scott Contini, Ron Steinfeld, Josef Pieprzyk and Krystian Matusiewicz,"A critical look at cryptographic hash function literature," Ecrypt Hash Workshop 2007 [13]

ハッシュ関数については、様々な定義や要件が存在するが、それらの多くは合致しない。このサーベイでは、様々な定義を議論し、この研究分野がどのように進化してきたか、また、今日人々がもつ研究目的を正確に描写することにより、本研究分野を整理するための初期検討を行っている。

Matthieu Finiasz, Philippe Gaborit and Nicolas Sendrier,"Improved fast syndrome based cryptographic hash function," Ecrypt Hash Workshop 2007 [24]

Mycrypt 会議で発表された証明可能安全な衝突耐性をもつハッシュ関数のファミリーを二つの方法で改良する方法を提案している。それらは、最終圧縮関数を追加し、出力長の半分に等しい前線性レベルを達成することと、ハッシュ関数に対する記述をより短くするため、汎用ランダム行列を用いる代わりに、ランダムな半巡回行列を用いることである。この短い記述は複数の観点で役に立つ。ひとつは、行列が標準的な CPU のキャッシュに適すること、他のひとつは、メモリーが限られた環境等の新しい適用が可能になることである。

J.J. Hoch, A. Shamir, Breaking the ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions, FSE 2006 [35]

ハッシュ関数のハッシュ長を伸長する方法として、複数の繰り返しハッシュ関数の出力を連結してハッシュ値を生成する方法がある。この方法で生成されるハッシュ関数を繰り返し連結 (IC) ハッシュ関数と呼ぶ。しかし、2004 年に Joux により、繰り返しハッシュ関数の多重衝突攻撃及び、それを用いた IC ハッシュ関数の衝突攻撃が発表され、IC ハッシュ関数はそのハッシュ長に見合う安全性を有さないことが分かっている。これに対し、IC ハッシュ関数の安全性を高めるために、メッセージブロックを複数回スキャンする

ことを許し、かつ、メッセージブロックの入力順を各圧縮関数毎に任意に並び変えられる、という拡張を施した繰り返し連結拡大 (ICE) ハッシュ関数を考えることができる。ICE ハッシュ関数は、メッセージブロックのスキャン回数が最大で 2 回という制約の下では、Nandi 等により、2005 年に解読されている。本論文では、スキャン回数を s 回以下と一般化した ICE に対しても、内部の繰り返しハッシュ関数に対して多重衝突攻撃が構成でき、それにより ICE に対する衝突攻撃が構成できることを示している。

Markku-Juhani O. Saarinen, "Linearization Attacks Against Syndrome Based Hashes," INDOCRYPT 2007 [88]

計算量理論の困難な問題に基づいて、安全性の証明を持ついくつかのハッシュ関数が提案されている。その中で、符号理論の困難な問題に基づいた関数として、Fast Syndrome Based Hash (FSB) が提唱されている。FSB は圧縮関数を提案しているが、本文献により、関数の原像および衝突が発見されている。攻撃は 128 ビットの安全性を持つと設計者の主張するアルゴリズムに対して、デスクトップ PC で 1 秒以内で行われている。本文献の著者は、この攻撃の存在の理由として、FSB が基づいている問題は平均的には困難であるが、特別な場合には簡単に解けるものであることを挙げている。

T. Ristenpart and T. Shrimpton, "How to Build a Hash Function from Any Collision-Resistant Function," ASIACRYPT 2007 [81]

計算量理論の困難な問題に基づいて、安全性証明が可能なハッシュ関数が提案されている。これらのハッシュ関数において扱われる安全性は多くの場合、衝突耐性に関するものであり、ハッシュ関数に期待される性質のうち衝突耐性以外ののいくつかを持ちあわせていない場合がある。特に、ランダムオラクルと容易に区別されてしまう場合がある。本論文では、ランダムオラクルとの区別のつかないハッシュ関数を衝突耐性のある関数から作ることを目指して、Mix-Compress-Mix (MCM) 構成という構成法を提案している。これは、衝突耐性を持つ関数 H を 2 つの単射な関数 ε_1, ε_2 で挟むというものである。

$$MCM(M) = \varepsilon_2 \circ H \circ \varepsilon_1(M). \tag{6}$$

ε_1, ε_2 が単射なランダムオラクルの場合、MCM が monolithic なランダムオラクルになることを証明している。

S. Hirose, J.H. Park, A. Yun, "A Simple Variant of the Merkle-Damgård Scheme with a Permutation," ASIACRYPT 2007 [34]

本論文では、新しい定義域拡大方式 Merkle-Damgård with a permutation (MDP) を提案し、安全性の証明を与えている。この方式は、MD 方式の変形版であり、最終メッセージブロックを処理する直前に内部状態が置換により変換することが特徴である。MD 方式は、メッセージを拡張することにより、正当なハッシュ値を計算できてしまうという望ましくない性質 extension property を持つが、MDP 方式はこの性質を持たない。安全性証明としては、PRF としての性質やランダムオラクルとしての性質を扱っている。

2.5 まとめ

証明可能なハッシュ関数の安全性評価技術について、調査と検討を行なった。ハッシュ関数の安全性評価を行うための基礎をなす安全性評価のためのモデルに関する動向を整理した上で、既存のハッシュ関数について、標準として利用されているもの、また、近年新しく提案されたものを中心としての比較を行なった。これらは、最近の解読技術に進展に伴い、以前よりその重要性が増している技術である。本章での調査と検討により、設計方法や安全性評価のためのモデル等に関し、抜本的な見直しが行われていることを把握し、ハッシュ関数の定義域拡大の構成方法や証明方法に関する技術や、本研究分野の方向性や課題に関する知見を得た。

3 実装性能重視型ハッシュ関数の安全性評価

3.1 調査目的

本章では、近年新たに提案されているハッシュ関数やその他の関連する共通鍵暗号技術の安全性と安全性評価手法の調査を行い、汎用的にこの安全性評価手法を活用するための検討を行う。

SHA-1 や MD5 などの実際に使われているハッシュ関数の多くは、圧縮関数という入力長が固定の関数を繰り返し適用することにより、任意長のメッセージを処理する。ここで、圧縮関数の繰り返し方法は、定義域拡大と呼ばれている。定義域拡大には、MD(Merkle-Damgaard) 構成法が一般的に用いられていたが、この構成法に対しては、近年、様々な攻撃が提案されており、この構成法についての見直しが検討され、新しい定義域拡大も提案されている。

本章では、ハッシュ関数を 2 つに分け考察する。一つは、上述したように定義域拡大と圧縮関数を組み合わせたものであり、一つは弱い攪拌関数をメッセージブロックの数と同じ段数重ねることにより衝突困難性を達成し、その後の最終処理により一方向性を達成するように設計されたものである。前者を定義域拡大+圧縮関数型、後者を PANAMA 型と呼ぶことにする。以下にこれらの二組の型についてステートへの入力幅の大小の観点から分類しそれらの代表例を示す。ステートへの入力幅を比較すると、初めに定義域拡大+圧縮関数型では Whirlpool は 100%であり、SHA-1 は 20%である。100%に近いと高速だが、メッセージ撹拌の部分を非線形にするなど重くする必要があるが、割合が小さいと低速になるがメッセージ撹拌部を線形など軽いものを用いることができる。次に PANAMA 型では割合は 50% である。

定義域拡大+圧縮関数型のハッシュ関数は、構造的に、いくつかの層に別れており、各層間で安全性の還元が行える等の利点がある。圧縮関数は、ブロック暗号をベースに構成されているものが一般的である。主なブロック暗号ベースハッシュ関数を表 5 に掲げ、その構成法や撹拌のために用いられている演算を比較する。

3.2 定義域拡大＋圧縮関数型ハッシュ関数の安全性評価

本節では、定義域拡大+圧縮関数型ハッシュ関数の安全性評価に関する調査報告をする。特に重点攻撃手法として、Kelsey 等による Tiger の攻撃、Saarinen による FORK-256 の攻撃を挙げる。Tiger の攻撃においては、ラウンド関数での中間一致によりメッセージ更新を行うという新しい手法が取られており、重要である。また、FORK-256 の攻撃においては、フルラウンドの FORK-256 が破られており、興味深い。その他、MD4、MD5、HAVAL、SHA-1、SHA-256 およびに関しての文献も紹介する。

表 4: 専用ハッシュ関数

ハッシュ関数	ハッシュ長 (bit)	圧縮関数用モード	基本構造
FORK-256	256	Davies-Meyer mode	定義域拡大+圧縮関数
Grijndahl	256 or 512	-	PANAMA 型
MD5	160	Davies-Meyer mode	定義域拡大+圧縮関数
MAME	256	Matyas-Meyer-Oseas mode	定義域拡大+圧縮関数
PANAMA	無制限	-	PANAMA 型
RADIOGATUN	無制限	-	PANAMA 型
RIPEMD	128	Davies-Meyer mode	定義域拡大+圧縮関数
RIPEMD-128	128	Davies-Meyer mode	定義域拡大+圧縮関数
RIPEMD-160	160	Davies-Meyer mode	定義域拡大+圧縮関数
Tiger	192	Davies-Meyer mode	定義域拡大+圧縮関数
SHA-1	160	Davies-Meyer mode	定義域拡大+圧縮関数
SHA-1-IME	160	Davies-Meyer mode	定義域拡大+圧縮関数
SHA-256	256	Davies-Meyer mode	定義域拡大+圧縮関数
SHA-512	512	Davies-Meyer mode	定義域拡大+圧縮関数
Whirlpool	512	Miyaguchi-Preneel mode	定義域拡大+圧縮関数

表 5: ハッシュ関数の構成ブロック暗号

ハッシュ関数	鍵スケジュールの線型性と演算	データ撹拌部における演算
FORK-256	線型：ワード単位置換	ブール関数
MD5	線型：ワード単位置換	ブール関数
RIPEMD	線型：ワード単位置換	ブール関数、算術加算、巡回シフト
RIPEMD-128	線型：ワード単位置換	ブール関数、算術加算、巡回シフト
RIPEMD-160	線型：ワード単位置換	ブール関数、算術加算、巡回シフト
SHA-1	線型： 排他的論理和、巡回シフト	ブール関数
SHA-1-IME	線型： 排他的論理和	ブール関数
SHA-256	非線型：算術加算演算、巡回シフト	ブール関数
SHA-512	非線型	ブール関数
MAME	非線型：4 ビット S-box、巡回シフト	4 ビット S-box、巡回シフト
Tiger	非線型：算術加算、演算、巡回シフト	8 ビット入力 64 ビット出力 S-box
Whirlpool	非線型：8 ビット S-box、MDS 行列	8-bit S-box、巡回シフト

3.2.1 Tiger

Tiger は、1996 年に Anderson と Biham によって FSE で発表された 64 ビット CPU 向けのハッシュ関数であり、192 ビットのハッシュ値を出力する。これまでのところ、Tiger のフルラウンドモデル (24 ラウンド) に対する衝突攻撃は知られていないが、ラウンド簡約モデルに対する衝突攻撃、及びフルラウンドモデルに対する疑似近似衝突攻撃は存在する。

まず、FSE 2006 において Kelsey 等により 16 及び 17 ラウンドに簡約されたモデルに対する衝突攻撃が発表された [42]。この際、20 ラウンド簡約モデルの疑似衝突攻撃も同時に発表されている。その後、INDOCRYPT 2006 において、Mendel 等により、19 ラウンドでの衝突及び 22 ラウンドでの疑似近似衝突に改良された [58]。さらに、ASIACRYPT 2007 において Mendel 等により、フルラウンドでの疑似近似衝突及び 23 ラウンドでの疑似衝突に改良されている [59]。これらの結果を表 6 に纏める。以下では、これらの結果の基礎となる Kelsey 等による 16 ラウンドの攻撃について概略を説明する。

表 6: ハッシュ関数 Tiger に対する攻撃の概要

ラウンド数	攻撃の種類	計算量	文献
Tiger-16	衝突攻撃	2^{44}	[42]
Tiger-19	衝突攻撃	2^{62} と 2^{69}	[58]
Tiger-19	疑似衝突攻撃	2^{44}	[58]
Tiger-21	疑似衝突攻撃	2^{66}	[58]
Tiger-23/128	疑似衝突攻撃	2^{44}	[58]
Tiger-23	疑似衝突攻撃	2^{47}	[59]
Tiger-20	疑似近似衝突攻撃	2^{48}	[42]
Tiger-21	疑似近似衝突攻撃	2^{44}	[58]
Tiger-22	疑似近似衝突攻撃	2^{44}	[58]
Tiger	疑似近似衝突攻撃	2^{47}	[59]

Tiger の仕様

定義域拡大　Merkle-Damgård 構成をとっており、メッセージはパディングの後、512 ビット単位のメッセージブロックに分割される。圧縮関数は 192 ビットの内部状態と 512 ビットのメッセージブロックの計 704 ビットを入力としてとり、192 ビットの内部状態を出力する。最終メッセージブロックを処理した直後の内部状態 192 ビットをハッシュ値として出力する。

圧縮関数　圧縮関数は計 24 ラウンドから成る。24 ラウンドは 8 ラウンド毎の 3 つのパスに分けられる。内部状態、メッセージブロックともに 64 ビット単位に分けられ、それぞれ (A, B, C)、$(X_0, \cdots, X_7)$ と表示する。各ラウンドでは (A, B, C) を 1 つの 64 ビットワード X_i を用いて更新する。第 i ラウンド終了時の内部状態を (A_i, B_i, C_i) とすると、第 i ラウンドでの変換は以下である (図 1 参照)。

$$A_i = (Bi - 1 + \mathrm{odd}(C_{i-1})) \times \mathrm{mult}, \tag{7}$$

$$B_i = C_{i-1} \oplus X_i, \tag{8}$$

$$C_i = A_{i-1} - \mathrm{even}(C_{i-1}), \tag{9}$$

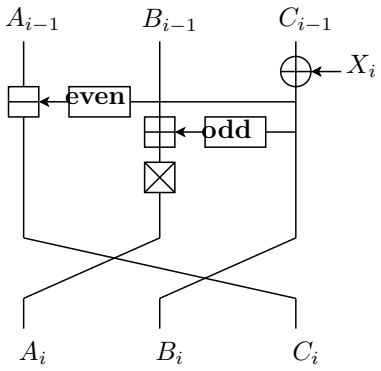

図 1: Tiger のラウンド関数

ただし、第 2 パス及び第 3 パスに使われる $X_8, \cdots, X_{23}$ は Key Schedule 変換により $X_0, \cdots, X_7$ より生成される。

$$(X_8, \cdots, X_{15}) = \mathrm{KeySchedule}(X_0, \cdots, X_7), \tag{10}$$

$$(X_{16}, \cdots, X_{23}) = \mathrm{KeySchedule}(X_8, \cdots, X_{15}), \tag{11}$$

KeySchedule 関数は、入力 $(Y_0, \cdots, Y_7)$ に対して、以下で定まる $(Y_0'', \cdots, Y_7'')$ を出力する。

first step	second step	
$Y_0' = Y_0 + (Y_7 \oplus \mathsf{A5A5A5A5A5A5A5A5})$,	$Y_0'' = Y_0' + Y_7'$,	(13)
$Y_1' = Y_1 \oplus Y_0$,	$Y_1'' = Y_1' - (Y_0' \oplus ((\neg Y_7') << 19))$,	(14)
$Y_2' = Y_2 + Y_1$,	$Y_2'' = Y_2' \oplus Y_1'$,	(15)
$Y_3' = Y_3 - (Y_2 \oplus ((\neg Y_2) << 19))$,	$Y_3'' = Y_3' + Y_2'$,	(16)
$Y_4' = Y_4 \oplus Y_3$,	$Y_4'' = Y_4' - (Y_3' \oplus ((\neg Y_2') >> 23))$,	(17)
$Y_5' = Y_5 + Y_4$,	$Y_5'' = Y_5' \oplus Y_4'$,	(18)
$Y_6' = Y_6 - (Y_5 \oplus ((\neg Y_4) >> 23))$,	$Y_6'' = Y_6' + Y_5'$,	(19)
$Y_7' = Y_7 \oplus Y_6$,	$Y_7'' = Y_7' - (Y_6' \oplus \mathsf{0123456789ABCDEF})$,	(20)

Tiger は 4 つの (8 ビット入力)-(64 ビット出力) の S-Box を用いる。(9), (7) の even(C), odd(C) はそれぞれ偶数、奇数番目のバイトを入力とした S-Box の出力で、次のように定義される。

$$\mathrm{even}(C) = T_1[c_0] \oplus T_2[c_2] \oplus T_3[c_4] \oplus T_4[c_6], \tag{21}$$

$$\mathrm{odd}(C) = T_4[c_1] \oplus T_3[c_3] \oplus T_2[c_5] \oplus T_1[c_7], \tag{22}$$

ここで、$T_1, \cdots, T_4$ は 4 つの S-Box で、c_j は C の j 番目のバイトである。

2-パスに簡約された Tiger に対する衝突攻撃

FSE2006 で Kelsey と Lucks により、16 ラウンドに簡約化された Tiger に対する衝突攻撃が発表された [42]。提案された攻撃の計算量は、2^{44} 回の圧縮関数呼び出しに相当する。

本攻撃では、64 ビット列 X, Y に対しては、XOR 差分が $\Delta_{\oplus}(X,Y) = 2^{63}$ の場合、算術差分も $\Delta_{+}(X,Y) = 2^{63}$ となり、両者が同じ値になるという性質を多用する。Tiger は算術演算、論理演算の双方を用いているが、この性質を用いることで部分的に両演算の差異がなくなり、攻撃が可能となる。以下では、この 64 ビット定数の差分値を $I = 2^{63}$ と記述する。

攻撃は鍵スケジュールの差分特性、ラウンド関数の差分特性、ラウンド関数でのメッセージ変更の 3 つの部分に分けられる。

鍵スケジュールの差分特性　16 ラウンド Tiger では、鍵スケジュールで、512 ビットメッセージ $X_0, \cdots, X_7$ を 1024 ビットに拡張する。1024 ビットのうち、最初の半分はメッセージそのもので、後ろの半分は $KeySchedule(X_0, \cdots, X_7)$ の出力 $(X_8, \cdots, X_{15})$ である。KeySchedule 関数は、入力差分 $(I, I, I, I, 0, 0, 0, 0)$ に対し、確率 1 で出力差分 $(I, I, 0, 0, 0, 0, 0, 0)$ を出力する。攻撃ではこの差分特性を用いる。

ラウンド関数の差分特性　ラウンド関数では 3 つの 64 ビット変数 (A, B, C) がラウンド毎に鍵を用いて更新される。16 ラウンド Tiger では、更新は全 16 段から成り、各ラウンドでは 64 ビット変数 X_i が使われる。ラウンド関数においては、第 6 ラウンドの入力時の内部変数の差分が $(I, I, 0)$ であり、かつ鍵スケジュールの差分が上述のものであれば、第 9 ラウンドでの出力差分は、確率 1 で $(0, 0, 0)$ となる。攻撃では、この差分特性を用いる。

上記 2 つの差分特性においては、第 9 ラウンドの終了時で内部状態の差分はなくなり、かつ、第 10 ラウンド以降の鍵スケジュールの差分もない。従って、ラウンドを進めていけば、出力されるハッシュ値は同一値となり、衝突ペアが得られる。そこで、攻撃として残る課題は、第 6 ラウンドの開始時に差分が $(I, I, 0)$ である内部状態を得ることである。

ラウンド関数での中間一致によるメッセージ更新　メッセージの自由度を用いて、目的の内部状態の差分を得ることを目指す。そのために次の局所的なメッセージ更新が重要となる。

ラウンド $i-1$ での内部状態 A, B, C と A', B', C' といくつかのメッセージ差分が与えられたときに、$i+1$ ラウンドでの C の算術差分 $\Delta^{+}(C_{i+1})$ が δ^* となるようなメッセージを計算量 2^{28} で求めることができる。具体的には、

$$\delta_{\mathbf{even}}(B_{i+1}, B^*_{i+1}) = (\Delta^{+}(B_{i-1}) + \delta_{\mathbf{odd}}(B_i, B^*_i)) \times (\text{const}) - \delta^*, \tag{23}$$

が満たされるように、X_i, X_{i+1} を決めればよい (図 2 参照)。$\delta_{\mathbf{even}}$, $\delta_{\mathbf{odd}}$ の各々に対し 2^{32} 回の計算が必要であり、圧縮関数にすると 2^{28} 回に相当する。

この局所的なメッセージ更新を用いて、大域的なメッセージ更新を行い、2^{44} 回の圧縮関数呼び出しで、第 6 ラウンド開始時の内部状態差分が $(I, I, 0)$ となるメッセージ対を得ることができる。従って、2^{44} 回の圧縮関数呼び出しでの衝突攻撃となる。

3.2.2 FORK

FORK-256 は FSE'06 で Hong 等により提案された 256 ビットのハッシュ値を出力するハッシュ関数である。FORK-256 に対しては、Matusiewicz 等 [53] 及び Mendel [57] 等により簡約モデルに対する衝突攻撃が見付けられ、さらに Matusiewicz 等 [56], [54] によりフルモデルに対する衝突攻撃が見付けられている。これを受け、開発者は既存の攻撃に耐性を持たせた改良版を 2007 年に発表した [38]。しかし、この改良版

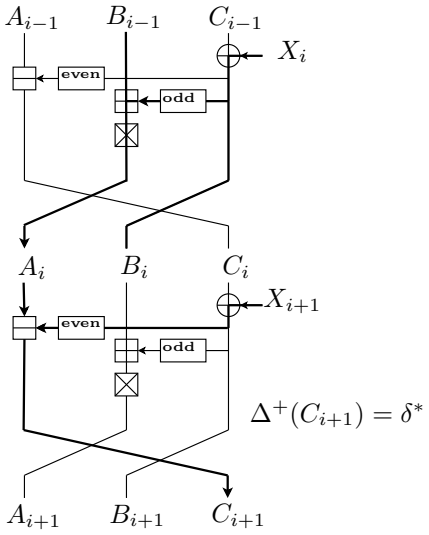

図 2: Outline of the message modification step in Tiger.

FORK-256 に対しても、INDOCRYPT '07 で Saarinen により衝突攻撃が発表されている [87]。以下では、この改良版 FORK-256 及びその衝突攻撃について見ていく。

FORK の仕様の概要

定義域拡大　FORK-256 は Merkle-Damgård 構成の 256 ビットハッシュ関数である。メッセージはパディングの後、512 ビット単位のメッセージブロックに分割される。内部状態は 256 ビットであり、1 ラウンドを構成する圧縮関数は、内部状態 256 ビットと 1 つのメッセージブロック 512 ビットの計 768 ビットを入力としてとり、更新された内部状態 256 ビットを出力する。最終メッセージブロックの処理が行われる最終ラウンドの圧縮関数の出力 256 ビットをハッシュ値として出力する。

圧縮関数　$i-1$ ブロック目終了時の内部状態を CV_i と記述する。i ブロック目において、圧縮関数は、内部状態 CV_i と i 番目のメッセージブロック M_i を入力として取り、内部状態 CV_{i+1} を出力する。以下では、メッセージブロック M_i を 16 個の 32 ビットワード $M_i[0], \cdots, M_i[15]$ で表わし、内部状態 CV_i を 8 個の 32 ビットワード $CV_i[0], \cdots, CV_i[7]$ で表わす。

圧縮関数は 4 本の独立な枝 $BRANCH_j$ $j = 1, \cdots, 4$ からなり、それぞれ CV_i と M_i を入力としてとる。各枝は、直列な 8 ステップからなり、各ステップでは 2 ワード分 (64 ビット分) のメッセージブロックと 2 ワード分の定数を用いて、256 ビットの内部状態を更新していく。各ステップでの更新後の内部状態を $R_j^{(s)} = (R_j^{(s)}[0], \cdots, R_j^{(s)}[7])$ と記述する。各 $R_j^{(s)}[k]$ はワードである。

$$R_j^{(s+1)} = STEP(R_j^{(s)}, M_i[\sigma_j(2s)], M_i[\sigma_j(2s+1)], \delta[\rho_j(2s)], \delta[\rho_j(2s+1)]). \tag{24}$$

ここで、σ_j 及び ρ_j は、それぞれ、ステップ毎に入力されるメッセージブロックのワード及び定数ワードを定める関数である。この式から分かるように、各枝、各ステップにおけるステップ関数は全て同一のものである。各枝の違いは、メッセージブロックのワード及び定数ワードの入力順 σ_j, ρ_j に集約されている。定

表 7: メッセージブロックの入力順

Step	枝 1		枝 2		枝 3		枝 4	
s	$\sigma_1(2s)$	$\sigma_1(2s+1)$	$\sigma_2(2s)$	$\sigma_2(2s+1)$	$\sigma_3(2s)$	$\sigma_3(2s+1)$	$\sigma_4(2s)$	$\sigma_4(2s+1)$
0	0	1	14	15	7	6	5	12
1	2	3	11	9	10	14	1	8
2	4	5	8	10	13	2	15	0
3	6	7	3	4	9	12	13	11
4	8	9	2	13	11	4	3	10
5	10	11	0	5	15	8	9	2
6	12	13	6	7	5	0	7	14
7	14	15	12	1	1	3	4	6

数ワードはメッセージブロックのワードと同様に、計 16 個ある。σ_j は以下の表 7 の通りである。定数ワードは後で述べる攻撃法で重要とならないため、ρ_j の記述は省略する。

ステップ関数は以下の通りである。

$$R_j^{(s+1)}[0] = R_j^{(s)}[7] \oplus (f(R_j^{(s)}[4] + b_j^{(s)} + \delta[\rho_j(2s+1)]) <<< 8), \tag{25}$$
$$R_j^{(s+1)}[1] = R_j^{(s)}[0] + a_j^{(s)} + \delta[\rho_j(2s)], \tag{26}$$
$$R_j^{(s+1)}[2] = R_j^{(s)}[1] + f(R_j^{(s)}[0] + a_j^{(s)}), \tag{27}$$
$$R_j^{(s+1)}[3] = R_j^{(s)}[2] + (f(R_j^{(s)}[0] + a_j^{(s)}) <<< 13) \oplus g(R_j^{(s)}[0] + a_j^{(s)} + \delta[\rho_j(2s)]), \tag{28}$$
$$R_j^{(s+1)}[4] = R_j^{(s)}[3] \oplus (g(R_j^{(s)}[0] + a_j^{(s)} + \delta[\rho_j(2s)])) <<< 17, \tag{29}$$
$$R_j^{(s+1)}[5] = R_j^{(s)}[4] + b_j^{(s)} + \delta[\rho_j(2s+1)], \tag{30}$$
$$R_j^{(s+1)}[6] = R_j^{(s)}[5] + g(R_j^{(s)}[4] + b_j^{(s)}), \tag{31}$$
$$R_j^{(s+1)}[7] = R_j^{(s)}[6] + (g(R_j^{(s)}[4] + b_j^{(s)}) <<< 3) \oplus f(R_j^{(s)}[4] + b_j^{(s)} + \delta[\rho_j(2s+1)]). \tag{32}$$

ただし、$a_j^{(s)} = M_i[\sigma_j(2s)]$, $b_j^{(s)} = M_i[\sigma_j(2s+1)]$ である。各枝は、最終ステップ ($s = 7$) 終了後の内部状態 $R_j^{(8)}$ を出力する。これら $R_j^{(8)}$ を用いて、圧縮関数の出力は、

$$CV_{i+1}[t] = CV_i[t] + ((R_1^{(8)}[t] + R_2^{(8)}[t]) \oplus (R_3^{(8)}[t] + R_4^{(8)}[t])), \quad t = 0, \cdots, 7, \tag{33}$$

と計算される。

改良版 FORK-256 に対する衝突攻撃

Saarinen により、改良された FORK-256 に対する衝突攻撃が INDOCRYPT '07 で発表された [87]。この攻撃は、一部のメッセージワードの拡散の不十分さを用いている。攻撃の概略は以下の通りである。

メッセージの拡散の不十分さ　1 つの枝では各 $M[k]$ は 8 ステップの間で a または b としてちょうど 1 回だけ使われる。特に、$M[1]$ は、枝 2 では $b_2^{(7)}$、枝 3 では $a_3^{(7)}$ として現れ、また、$M[14]$ は、枝 1 では $a_1^{(7)}$、

枝 4 では $b_4^{(6)}$ として現れる。最終ステップ付近で用いられたこれらのメッセージは十分拡散されない。実際に、更新関係式 (30), (29) から、

$$R_j^{(8)}[5] = R_j^{(7)}[4] + b_j^{(7)}, \tag{34}$$

$$R_j^{(7)}[4] = R_j^{(6)}[3] \oplus (g(R_j^{(6)}[0] + a_j^{(6)}) <<< 17), \tag{35}$$

となるが、これから $R_j^{(8)}[5]$ は $b_j^{(6)}$, $a_j^{(7)}$ には依存せず、$b_j^{(7)}$ には線形に依存することが分かる。従って、$R_1^{(8)}[5]$ および $R_4^{(8)}$ は $M[14]$ に依存せず、$R_3^{(8)}[5]$ は $M[1]$ に依存せず、$R_2^{(8)}[5]$ は $M[1]$ に線形に依存している。

ハッシュ値の部分固定　内部状態 (ハッシュ値) の一部を固定し、取り得るハッシュ値の空間を狭めることができれば、狭められた空間でバースデイ攻撃をすることにより、衝突を効率的に見付けることができる。例えば以下で考える $CV_0[5] = CV_1[5]$ のもとでは、224 ビットの空間での衝突を見付ければよく、計算時間は、$\sqrt{\frac{\pi}{2}} \times 2^{\frac{224}{2}}$ となる。以下では、これが実際に可能であることを見る。

衝突攻撃　$CV_0[5] = CV_1[5]$ を書き直すと、

$$R_2^{(8)}[5] - R_3^{(8)}[5] = R_1^{(8)}[5] - R_4^{(8)}[5], \tag{36}$$

となる。以下の 2 つの段階を経ることで、上式は達成される。

1. $M[1] = 0$ とし、$M[14] = 0, 1, \cdots, 2^{32} - 1$ に対して、枝 2 と枝 3 を計算する。ただし、$t = 1, 14$ 以外の $M[t]$ は適当に固定した値とする。各 $M[14]$ に対して、$x = R_2^{(8)}[5] - R_3^{(8)}[5]$ を計算し、$M[14]$ に対する x の対応表を作成する。

2. $t = 1, 14$ 以外の $M[t]$ を段階 1 での値に固定し、$M[1] = 0, 1, \cdots, 2^{32} - 1$ に対して枝 1 と枝 4 を計算し、$y = R_1^{(8)}[5] - R_4^{(8)}[5] + M[1]$ を求める。y は $M[14]$ には依存しないため、$M[14]$ を固定する必要はない。また、y に $M[1]$ を含めるのは x は $R_2^{(8)}[5]$ を通じて $M[1]$ に線形に依存しているためである。$y = x$ なる x が段階 1 の表に存在する場合、対応する $M[t]$ の組は式 (36) を満たす。

段階 1, 2 では、それぞれにおいて枝 2 つ分の計算しかしないため、各段階での計算量は、2^{31} 回の圧縮関数呼び出しとなり、計 2^{32} 回の圧縮関数呼び出しとなる。$M[14]$, $M[14]$, x, y はどれも 32 ビットなので、$x = y$ となる回数の期待値は 2^{32} である。段階 1 では y 以外の内部状態を保存していないとすると、$x = y$ となった場合に、再び枝 2 と枝 3 を計算する必要があり、2^{31} 回の圧縮関数呼び出しが追加で必要となる。結局、(36) を満たすメッセージとそのハッシュ値の組を 2^{32} 個得るために、$\frac{3}{2} \times 2^{32}$ 回の圧縮関数呼び出しが必要となり、メッセージ 1 個当りに換算すると $\frac{3}{2}$ 回の圧縮関数呼び出しとなる。従って、衝突に必要となる $\sqrt{\frac{\pi}{2}}2^{112}$ 個のメッセージとハッシュ値の組を得るためには、$\frac{3}{2} \times \sqrt{\frac{\pi}{2}} \times 2^{112} \approx 2^{112.9}$ の圧縮関数呼び出しが必要となる。なお、必要とされるメモリの大きさも時間と同じく $2^{112.9}$ である。

本攻撃法は、改良前の FORK-256 に対しても適用可能であり、ここでも $2^{112.9}$ 回の圧縮関数呼び出しで衝突を見付けることができる。

3.2.3 Tigerと FORK 以外のハッシュ関数

Magnus Daum,"Finding Differential Patterns for the Wang Attack," Conference Hash Functions [16]

Wang の攻撃の差分特性を探す部分とメッセージを修正して、その差分確率を改良する部分の二つの部分からなるが、本論文では、差分特性を探索部分に焦点を合わせている。Wang は、" 直感的に " や " 手で " でこの部分を行ったと言っているが、実際、この攻撃中で何が起こっているかを見てみることで、いくつかのアイデアが再構築できる。ここでは、MD5 に対する攻撃を考察している。二つのメッセージブロックを適用しているが、一つのメッセージブロックで near-collision を構成し、二番目のメッセージブロックで、衝突を構成している。ここでは、初めの部分の near-collision 探索は次のように行われている：Step1: 中間から最後のステップから構成される差分パターンの構築を行う。
Step2: MD5 の設計から、ステップ 33 からステップ 64 からなる near-collision が構成できる。
Step3: この差分伝播が高い確率で起こるような入力差分を選択する。
Step4: 最初の 32 ステップで差分伝播を実現できるようなレジスタの値を探索する。

Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen,"On Variable Bit-Rotations in SHA-1-like Hash Functions," Conference Hash Functions [77]

ハッシュ関数 SHA-1 はステップ関数の中で、constant bit-rotation を用いている。しかし、MD4 族の他のほとんどのハッシュ関数は、ステップによって rotation を行うビット数が変化する variable bit-rotation を用いている。今までのところ、これらの任意のハッシュ関数において、constant/variable bit-rotation 設計指針は与えられていない。本論文では、繰り返しハッシュ関数における variable bit-rotation を解析している。SHA-1 型のハッシュ関数に焦点を置き、これらの解析における最近の研究の観点から、constant bit-rotation の代わりに、variable bit-rotation を使用したときの利点や違いについて考察している。また、SHA-0 の設計の過程で、恐らく用いられたと思われる設計指針に関する洞察も与えている。

Hirotaka Yoshida, Alex Biryukov, and Bart Preneel, "Some applications of the Biham-Chen attack," Conference Hash Functions [96]

Crypt2004 で発表された Biham の攻撃に関して考察している。　対象とする安全性は、擬似衝突困難性と乱数性であり、これらがハッシュ関数のコア関数である圧縮関数に及ぼす影響について考究している。本稿での技術の有効性を検証するため、ハッシュ関数 MD5 の安全性評価を行っている。

Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen,"Breaking a New Hash Function Design Strategy Called SMASH," Conference Hash Functions [78]

MD4 族の構造に頼らないような新しいハッシュ関数の設計指針として SMASH があるが、本論文では SMASH に対して衝突攻撃を発表している。本方法では、ほぼ自由自在にハッシュ関数の中間値差分を作ることができる。秘密鍵が使用されないことにより、確率 1 で差分をつくることができる。衝突メッセージに対する束縛条件が少ないことを利用して、意味のあるメッセージで衝突を起こすことも成功している。衝突攻撃に必要なリソースは無視できるものであり、SMASH の設計指針に従う全てのハッシュ関数に対して、攻撃方法は働くと予想している。

Daniel J. Bernstein,"What output size resists collisions in a xor of independent expansions?," Ecrypt Hash Workshop 2007 [5]

各 $f_i(m_i)$ が長さ $3b$ ビットの入力を長さ $4b$ ビットの出力に拡張するときに、$m_1, m_2, m_3, m_4 -> f_1(m_1) + f_2(m_2) + f_3(m_3) + f_4(m_4)$(入力長は 12b ビット, 出力長は $4b$ ビット) という排他的論理和で構成される圧縮関数はどの程度安全かという問題を考察している。Wagner の一般化誕生日アルゴリズムを改良し、計算量を $2b$ から $24b/9$ に削減している。f_1, f_2, f_3, f_4 の選択の際に、ストリーム暗号 Salsa20 の構成要素を再利用することにより Rumba20 という圧縮関数を開発している。

Pierre-Alain Fouque, Gaetan Leurent and Phong Nguyen,"Automatic search of differential path in MD4," Ecrypt Hash Workshop 2007 [25]

MD4 の差分パスに焦点を置いている。パスに対する新しい内部表現に基づき、差分パス探索の新しい方法を与えている。この探索アルゴリズムを適用し、MD4 の衝突に対して、また、弱いメッセージに対する第二原像攻撃に対して、より良いパスを発見している。より正確には、新しいパスは衝突攻撃に関し、圧縮関数の各 3 ラウンドにおいて、新しいパスはより少ない条件を導く。これは、第二原像攻撃に対しても適用できる。

Antoine Joux and Thomas Peyrin,"Hash functions and the (amplified) boomerang attack," Ecrypt Hash Workshop 2007 [41]

Biham と Chen により、ハッシュ関数の攻撃方法として、ニュートラルビットの概念が提案された。本論文では、ブロック暗号解析のツールであるブーメラン攻撃をニュートラルビットのツールとして用いることにより、SHA-1 に対する計算量を削減することができることを示している。

Christophe De Canniere and Florian Mendel and Christian Rechberger,"On the full cost of collision search for SHA-1," Ecrypt Hash Workshop 2007 [18]

SHA-1 等における高速衝突探索はその多様性のため、それらを比較することが困難になっている。本論文では、様々な技術を調査し、比較を促進するために、単純だが効果的な方法を提案している。ケーススタディにおいて、攻撃の詳細を記述し、80 段から成る SHA-1 の 70 段に対して、新しく改善された衝突探索方法に対し、計算量の見積もりとパフォーマンスの測定法を示している。

J. Black, M. Cochran, T. Highland, "A Study of the MD5 Attacks: Insights an Improvements," FSE 2006 [10]

2004 年に Wang 等によって、MD5 の衝突が発見された。本論文では、Wang 等による攻撃の深い理解を目指し、多重メッセージ変更の方法を説明し、また、差分パスの探索方法に関する洞察を与えている。これらを用いて、衝突攻撃の速度を上げることに成功しており、衝突探索を実装したツールを公開している。Wang 等が IBM のスーパーコンピュータを用いて約 1 時間で求めたものを、本論文の方法によれば、普及型 PC により 11 分で求めている。

F. Mendel, N. Pramstaller, C. Rechberger, V. Rijmen, "Analysis of Step-Reduced SHA-256," FSE 2006 [60]

本論文では、Wang 等の衝突攻撃に対する SHA-256 の安全性を解析し、SHA-256 のメッセージ拡張は、Wang 等の攻撃の適用を妨げるものとなっていることを示している。これを克服するために、パータベーションベクトルが正当な拡大メッセージになるという条件を落とすことを考案し、ベクトルの探索方法を開発している。このパータベーションベクトルを用いることで、高い確率で衝突を生成する特性を見付けることができる。条件を落とした代償として、パータベーションベクトルの探索空間が大きくなることが挙げられるが、この大きさを削減する方法も述べられている。この方法を用いて、18 ステップに簡約された SHA-256 に対する衝突、および、22 ステップに簡約された SHA-256 に対する疑似衝突を汎用 PC において見付けている。

F. Mendel, N. Pramstaller, C. Rechberger, V. Rijmen, "The Impact of Carries on the Complexity of Collision Attacks on SHA-1," FSE 2006 [61]

本論文では、Wang 等の SHA-1 の衝突攻撃にかかる計算量を以前より正確に計算している。この攻撃の計算量は、線形化された SHA-1 における 6 ステップの局所衝突の生起確率に多きく依存する。従来の見積もりでは、実際の SHA-1 で局所衝突が起きるための条件を求め、この条件と上記確率を基に計算量が予想されている。本論文では、実際の SHA-1 で局所衝突が生起する条件の変わりに、生起する確率を用いることで、より正確に計算量が計算できることを指摘している。この方法の計算によれば、以前に見積もられていた計算量よりも少ない計算量で攻撃が成功することが分かる。

S. Indesteege, B. Preneel, "Preimages for Reduced-Round Tiger," WEWoRC 2007 [39]

本論文は、ラウンドが削減された Tiger に対する原像攻撃を記述している。Mendel 等が与えた 3 ラウンド分の状態更新変換での原像の解法 [58] を用いて、12 ラウンドに削減された圧縮関数の原像を、$2^{63.5}$ 回の圧縮関数呼び出しで求める方法を与えている。これを用いて、ラウンドが削減された Tiger に対して、第二原像攻撃及び原像攻撃を構成しており、計算量はそれぞれ $2^{63.5}$, $2^{64.5}$ 回の圧縮関数呼び出しとなっている。13 ラウンドに削減された Tiger の攻撃にも拡張しており、この場合、第二原像および原像攻撃の計算量は、それぞれ $2^{127.5}$, $2^{128.5}$ 回の圧縮関数呼び出しとなっている。

M. Schläffer, E. Oswald, "Searching fo Differential Paths in MD4," FSE 2006, [89]

Wang 等の一連のハッシュ関数への攻撃のオリジナル論文では、差分パスおよび条件式の決定方法に関して詳しい記載はなかった。そのような中、攻撃に関するさらなる洞察を得るために、本論文では、MD4 に対して Wang 等の攻撃で用いる差分パスの自動探索アルゴリズムを提案している。MD4 の最初の 24 ステップに対する差分パスを 1000 以上求め、その中で最良のパスを示している。Wang 等のパスに比べ、攻撃の第二ラウンドでの条件式の数が少なく、このパスを用いた攻撃は Wang 等のものよりも計算量が少ないものになっている。

H. Yu, X. Wang, A. Yun, S. Park, "Cryptanalysis of the Full HAVAL with 4 and 5 Passes," FSE 2006 [97]

本論文は、4 パスおよび 5 パス HAVAL に対する衝突攻撃を記述している。この攻撃は、まず適切なメッセージのモジュラー差分とその差分に沿った差分パスを求める。そして、内部状態に対する条件式を決定し、条件式が満たされるようにメッセージ更新を行う。この方法により 4 パス HAVAL に対して、2 つの現実的な衝突攻撃を見付けている。2 つの攻撃ともメッセージペアは 2 ブロックメッセージ (2048 ビット) のペアである。1 つは、各ブロックの 1 ワードのみに差分があるものであり、2^{43} 回のハッシュ関数呼び出しの計算量で衝突が行える。もう 1 つは、各ブロックの 2 ワードに差分があるもので、計算量は 2^{36} 回のハッシュ関数呼び出しである。さらに、本論文では 5 パス HAVAL に対して、計算量が 2^{123} 回のハッシュ関数呼び出しとなる、理論上の衝突攻撃も提案している。

3.3 PANAMA 型ハッシュ関数の安全性評価

バッファ、ステートという二つの型の内部状態をもつハッシュ関数を PANAMA 型ハッシュ関数と呼ぶことにする。全ての内部状態は 0 にセットされる。各メッセージブロック対して次のステップが実行される。初めにステートは非線形変換を適用することにより更新される。次にバッファ一部とメッセージがステートに排他的論理和される。3 番目にメッセージブロックがバッファに入力される。バッファの内容が線形変換により更新される。PANAMA 型ハッシュは、PULL モードと PUSH モードからなり、PUSH モードでは、バッファにメッセージ入力があり、その PUSH モードでは衝突困難性を実現し PULL モードでは一方向性を実現しているという点で、Whirlpool, SHA-1 など個々のハッシュ関数に専用に設計されたブロック暗号で攪拌を行い、その後の安全性の証明されたモードで一方向性を実現しているものとは異なった構造をもつ。

近年、PANAMA 型ハッシュ関数の提案されつつあり、このタイプのハッシュ関数のいくつかに対し、パフォーマンスの観点での利点が報告されているが、安全性の観点からは、評価手法や攻撃手法の成熟度は高いとはいえないのが現状である。このタイプのハッシュ関数で主要なものは、PANAMA, RADIOGATUN, Grijndael 等である。

3.3.1 PANAMA に対する攻撃

PANAMA は、ハードウェアや IC カード実装において実装規模が大きくなってしまうという欠点があるが、ソフトウェア実装においては、かなり良いパフォーマンスを示している。PANAMA はメッセージスケジュールの役割をしているのは、バッファと呼ばれる線形フィードバックレジスタである。

PANAMA は、Rijmen らによって解読された。PANAMA のハッシュ値は 256 ビットだが、彼らの解読方法では計算量は 2^{82} であり、誕生日攻撃の計算量 2^{128} より大幅に少ない計算量で衝突を見つけることができる。攻撃方法は、差分攻撃によるもので、メッセージ差分を入力し、あるブロック処理後にバッファ、ステート双方の差分を消滅させるという方法である。実際には、ステートの衝突はそのいくつかの部分衝突に分けて考察し、それぞれの部分衝突における差分を消滅させるという方法を取る。まず、バッファに対する差分伝播を表す衝突のフォーマットを求め、それを用いてステートの衝突フォーマット見つける。次に、ステートにはメッセージ入力の他にバッファからの入力が入るので、メッセージ差分が一度入力されると、その数ブロック後に、もう一度その差分値がバッファ入力としてステートに入力される。しかし、この影響により踏まえたステートの差分伝播フォーマットを探索は困難になりそうだが、実際には困難ではなく見つ

けられている。差分フォーマットは、ステートの攪拌関数 ρ が非線形変換なので即値がある条件をみたすときに限って満たされる。このフォーマットから直ちに差分伝播に満たす即値の方程式が導出される。この方程式が可解であるか否かは、その前のステートに入るメッセージ入力に依存する。攻撃者は、メッセージの即値をコントロールすることにより可解であるような方程式を探索する。このときに費やす時間が攻撃に必要な計算量である。この計算量が誕生日攻撃の計算量より大幅に小さいことが発見された。PANAMA がこの攻撃で解読可能なことの原因として、PANAMA では一度のバッファ入力の際に、約半分という大きな割合でステートへメッセージが入力されることと、ρ の非線形部が暗号学的に非常に弱いことが考えられる。このことは、攻撃者が差分や即値をコントロールする際の自由度が大きいことを攻撃可能なことを意味する。

3.3.2 Grindahl に対する衝突攻撃

本節では、Grindahl への衝突攻撃に関して記述する。本攻撃法は、差分の少ない状態ではなく、差分が全てのバイトにある状態を出発点としたパスを用いている点において、興味深い。

Grindahl の仕様

Grindahl は FSE2007 で Knudsen らによって提案されたハッシュ関数である [49]。Grindahl-256 と Grindahl-512 が提案されており、それぞれハッシュ値は 256 ビット、512 ビットとなっている。以下では、Grindahl-256 について記述するが、設計の大枠は、512 ビット版でも同じである。

基本設計　Grindahl-256 は、ビット長 $m = 384$ ビットの内部状態を持つ。メッセージ d を、$d = d_1||d_2||\cdots||d_t$ とビット長 $b = 32$ ビットのメッセージブロック d_i に分割し、ラウンド毎にメッセージブロック d_i を用いて、内部状態を更新していく。i ラウンド目終了時の内部状態を s_i と書く。第 1 ラウンド開始時の内部状態は、s_0 と書き、事前に定義した値とする。ラウンド関数は”Concatenate-Permute-Truncate” と呼ばれるデザインに基づいており、各ラウンドは Concatenate, Permute, Truncate という 3 つの段階からなる。

$$S_i \leftarrow d_i||s_{i-1} \qquad \text{(Concatenate)}, \tag{37}$$

$$\hat{S}_i \leftarrow P(S_i) \qquad \text{(Permute)}, \tag{38}$$

$$s_i \leftarrow \mathrm{trunc}_m(\hat{S}_i) \qquad \text{(Truncate)}, \tag{39}$$

ここで、$S_i, \hat{S}_i$ は $m + b = 416$ ビット長であり、拡大内部状態と呼ぶ。また、P は $\{0,1\}^{m+b}$ から $\{0,1\}^{m+b}$ への非線形な全単射の関数であり、trunc_m は入力の最下位 m ビットを出力する関数である。

最終メッセージブロックまで処理し終わったら、次の $\nu_{\mathrm{br}} = 8$ ラウンドのブランクラウンドが行われる。

$$\hat{S}_i \leftarrow P(\hat{S}_{i-1}), \quad i = t+1, \cdots, t+\nu_{\mathrm{br}}, \qquad \text{(Blanck round)} \tag{40}$$

最後に、最終拡大内部状態 $\hat{S}_{t+\nu_{\mathrm{br}}}$ の最下位 $n = 256$ ビット $\mathrm{trunc}_n(\hat{S}_{t+\nu_{\mathrm{br}}})$ をハッシュ値として出力する。

非線形全単射関数 P の設計　Grindahl は非線形全単射関数として Rijndael の設計を利用している。Rijndael が用いている置換関数は差分解読に強いことが知られており、ハッシュ関数で用いても、その恩恵を受ける

ことができると考えられる。Rijndael と同様に、ビット列を各要素が 1 バイトから成る行列で扱う。拡大内部状態 S_i を $x_0||x_1||\cdots||x_{51}$ とバイト x_i で表示した場合、$\alpha_{i,j} = x_{i+4*j}$ として、

$$S_i = \begin{pmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,12} \\ \alpha_{1,0} & \cdots & & \vdots \\ \alpha_{2,0} & \cdots & & \\ \alpha_{3,0} & \cdots & & \alpha_{3,12} \end{pmatrix}, \tag{41}$$

と記述する。この行列に対して、以下の置換 P が作用する。

$$P(S_i) = \mathsf{MixColumns} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes} \circ \mathsf{AddConstant}(S_i), \tag{42}$$

MixColumns, SubBytes は Rijndael と同じものであり、ShiftRows はシフトする量を第 1 行から順に $1, 2, 4, 10$ としたもの (AES では $0, 3, 2, 1$) である。AddConstant は、定数行列を各成分毎に排他的論理和するものである。

なお、行列表示では、第 $i-1$ ラウンドの Truncate と第 i ラウンドの Concatenate を合成した作用は、$\hat{S}_{i-1}$ の第 1 列を d_i の 4 バイトで置き換えることに他ならない。

設計者による安全性の主張　Grindahl-256 は、原像耐性、第二原像耐性および衝突耐性の全てに対して、$2^{\frac{256}{2}}$ の安全性を持つと、設計者は主張している。

Grindahl に対する切り詰め差分攻撃

Grindahl は、内部関数に AES の非線形な置換を用いており、具体的な差分値を用いる差分攻撃において高確率で成立する差分特性を見付けることは難しいと考えられる。しかし、差分の存在の有無のみを用いる切り詰め差分攻撃を用いれば、置換の線形部のみに着目することが可能となり、攻撃可能となることが、Peyrin により ASIACRYPT '07 で示された [75]。以下、この攻撃について記述する。

衝突攻撃としては、ハッシュ値の出力の際に衝突を起こす外部衝突と内部状態の段階で衝突を起こす内部衝突が考えられる。外部衝突では、Blanck round を考慮する必要がでてくるが、この round ではメッセージブロックの入力が無いため状態の制御が難しい上に、差分特性に対する性質の良い置換が用いられているために、攻撃をするのは難しいと考えられる。このため、本攻撃では、Blanck round 直前での拡大内部状態の衝突を目指している。

本攻撃では、MixColumns の差分伝播性とメッセージブロックの内部状態への攪拌における時間差が重要となる。まず、これらの性質を見ていく。

MixColumns の差分伝播性　MixColumns は $\{0,1\}^{4\times 8}$ から $\{0,1\}^{4\times 8}$ への線形写像である。入力のペア V_1, V_2 に対して、出力をそれぞれ W_1, W_2 とする。入力差分 $\Delta V = V_1 \oplus V_2$ のちょうど D_I 個の特定のバイトが 0 でないの場合に、出力差分 $\Delta W = W_1 \oplus W_2$ のちょうど D_O 個の特定のバイトが 0 でない確率は、近似的に次の通りである。

D_O \ D_I	0	1	2	3	4
0	1	0	0	0	0
1	0	0	0	0	1
2	0	0	0	2^{-8}	1
3	0	0	2^{-16}	2^{-8}	1
4	0	2^{-24}	2^{-16}	2^{-8}	1

MixColumns は、入出力の 0 でない差分バイトの最小数 (最大差分伝播) が 5 であり、上表はこのことを反映している。例えば、$D_I = 2$, $D_O = 2$ のエントリーは 0 となっている。

メッセージブロックの内部状態への攪拌　メッセージブロックは Concatenate により、拡大内部状態に組込まれ、ShiftRows と MixColumns により、内部状態へ攪拌される。この攪拌は非常に速く進むが、1 ラウンドで全ての内部状態のバイトに影響が及ぶわけではない。ShiftRows のシフトパラメータが 1, 2, 4, 10 であることから、1 ラウンド目では、メッセージバイトの 1, 2, 3, 4 バイト目は、それぞれ順に内部状態の 1 列目, 2 列目, 4 列目, 10 列目のみにしか影響を与えない。2, 3 ラウンド目では攪拌が進み、より多くの列に影響を与えるが、各バイトで見ると尚も全列に影響を与えるわけではない。4 ラウンドで始めて、各バイトはそれぞれ全内部状態に影響を及ぼす。このように攪拌に時間がかかるため、メッセージバイトを調節することによりそのラウンドの内部状態だけでなく、数ラウンド先の内部状態まで望みの形にすることができる。つまり、効率良くメッセージバイトを攻撃に用いることができる。

それでは、切り詰め差分攻撃の大枠を見ていく。

切り詰め差分パス　一般には、切り詰め差分パスは、差分のあるバイトの数がパス上で常に少ない場合に成立確率が高くなると考えられるため、このようなパスを見付けることが良いと思われる。しかし、Grindahl の設計者による評価により、一度差分を持ち込むとその差分は非常に多くのバイトに伝播することが確かめられているため、この方向性での攻撃法探索は難しいと考えられる。一方、全バイトが差分を持つ状態は非常に簡単に得られるため、本攻撃の提案者は、この状態を出発点として選択している。提案されている切り詰め差分パスは、全バイトに差分がある状態から全バイトに差分がなくなるまでのパスであり、8 ラウンドに及ぶ。同じ始状態から終状態に行くパスとして 4 ラウンドしかかからないパスも存在するが、より多くのメッセージバイトを攻撃に用いることのできる 8 ラウンドのパスの方が、攻撃計算量が少ない。

切り詰め差分パスは拡大内部状態の各バイトでの差分の有無をラウンド毎に追ったものである。ラウンド更新では、まず、Concatenate においてメッセージブロックが第 1 列に上書きされ、ShiftRows で各行がシフトパラメータに従ってシフトされる。最後に MixColumns により各列が変換され、次のラウンドに移る。切り詰め差分攻撃に特徴的なことは、バイトの差分の有無のみを見ているため、バイト毎の変換である SubBytes は切り詰め差分パスの段階では考慮する必要がないことである。

パスは、メッセージブロックでの差分の有無と MixColumns の出力での差分の有無を決定することにより決まる。メッセージブロックはもともと攻撃者が勝手に決められるものなので、差分の有無をどのように決めても計算量は増えない。一方、MixColumns の出力差分の有無は、MixColumns の差分伝播性の段落で書いた成立確率で出現するため、計算量に直接効いてくる。上記 8 ラウンドのパスでは、この確率を単純に全て掛けあわせると 2^{-440} となり、バースデイ攻撃に比べて成功確率が低く、有効な攻撃となっていない。しかし、メッセージブロックを調節して MixColumns の出力差分を制御することが可能であり、攻撃確率を

上げることができる。特に、前述したように、メッセージブロックが内部状態に完全に影響を及ぼすまでには 4 ラウンド要するため、MixColumns の出力差分の制御に、それ以前の数ラウンドのメッセージブロックからのバイトを用いることができ、多くの MixColumns の出力差分を制御できる。結果として攻撃確率を大幅に向上できる。このメッセージブロックによる制御を考慮に入れると、前述の 8 ラウンドの切り詰め差分パスに沿って、2^{-112} の確率で成功する攻撃が可能となる。この攻撃の主要な計算は初期状態を生成するところにあるため、2^{112} 組の初期状態 (全てのバイトに差分がある状態) を作る計算量で成功する攻撃となっている。

本攻撃の回避方法　本攻撃の提案者は、本攻撃を回避する方法の一つとして、内部状態を大きくとることを提案している。ただし、どの程度の大きさにすればよいか等の具体的な提案はしていない。

3.4 まとめ

本節では、実装性能重視型のハッシュ関数の安全性評価技術について、調査と検討を行なった。特に、近年盛んに研究がなされてきた、衝突耐性に対する攻撃を重点的に扱った。既存のハッシュ関数を、定義域拡大+圧縮関数型と PANAMA 型の 2 つに分け、それぞれにおいて、具体的なアルゴリズムに対する攻撃例を調査、検討した。

定義域拡大+圧縮関数型では、ラウンド削減された Tiger 及びフルラウンドの FORK-256 に対する衝突攻撃を扱った。簡約版 Tiger に対しては、中間一致によるメッセージ更新の手法を用いて差分解析で攻撃されることを見た。攻撃は、16 ラウンドの Tiger に対して、2^{44} 回の圧縮関数呼び出しで行われるものである。新たなハッシュ関数の設計の際には、この手法に対する耐性も考える必要があることが分かった。FORK-256 に対しては、メッセージの拡散の不十分さに起因して、フルラウンドの関数が破られることを見た。攻撃計算量は、$2^{112.9}$ 回の圧縮関数呼び出しである。FORK-256 の圧縮関数は、4 並列で処理を進めるという実装効率上の利点があるが、逆にこの点が安全性上の弱点となり、攻撃を可能にしたと言える。その他のアルゴリズムとして、MD4, MD5, SHA-1, SHA-256, HAVAL 等の調査、検討を行った。

PANAMA 型としては、PANAMA 及び Grindahl を扱った。特に、Grindahl の衝突耐性に対する切り詰め差分攻撃に関して詳しく調査、検討した。攻撃は、全バイトに差分のある状態ペアを 2^{112} 組だけ生成する計算量で行われる。Grindahl は Rijndael の構成要素を用いて構成されており、MixColumns 及び ShiftRows の効用により、わずかな差分が数ラウンドのうちに全バイトに広がるという性質を持つ。しかし、攻撃では、この性質を逆手にとり、全バイトに差分がある状態を出発点にしている。設計者は、差分のあるバイトが常に少ないような差分パスは存在しないことは確かめていたが、攻撃手法のパスはこのようなパスには含まれない。このことは、設計者は一般的な攻撃法のみならず、多様な攻撃法をできる限り想定する必要があることを、示している。

4 共通鍵暗号をベースとしたハッシュ関数の安全性の評価ツール仕様の検討

4.1 調査目的

We here attempt to give an approach to develop a tool analyzing 256-bit block-cipher based hash functions regarding differential cryptanalysis.

4.2 検討対象

An approach to develop a tool analyzing 256-bit block-cipher based hash functions regarding differential cryptanalysis.

4.3 検討項目

4.3.1 コンセプト

One could construct a hash function from a block cipher using the Davies-Meyer construction. Inversely, one can construct a block cipher which is the compression function with Davies-Meyer chaining peeled off. In the cipher, the message block M_j is viewed as the key, the chaining variable V_j acts as the plaintext block and $V_{j+1} = E(V_j, M_j)$ is the corresponding ciphertext block. In general, the block cipher constructed from a hash function in such a way is called a hash function in encryption mode.

Several cryptanalytic techniques ranging from differential cryptanalysis [7] to slide attacks [8] have been applied to study the security of well-known hash functions in encryption mode. For example, differential cryptanalysis of SHA-1 has been shown in [30]. A slide attack on SHA-1 and an attack on MD5 which finds one high-probability differential characteristic were given in [84].

The technique of differential cryptanalysis has first been described in [7]. The aim of the approach is to find differential characteristics for the whole cipher. In [7], a differential characteristic is defined in the following:

Definition 9 *Associated with any pair of encryptions are the difference of its two plaintexts, the differences of its ciphertexts, the differences of the inputs of each round in the two executions and the differences of the outputs of each round in the two executions. These differences form an n-round characteristic. A characteristic has a probability, which is the probability that a random pair with the chosen plaintext difference has the round and ciphertext differences specified in the characteristic.*

Hereafter we will use the notion of a “round”, as defined in the specification of block ciphers. We will also use the notion of a “pass”, which stands for 32 rounds.

We will analyze a 256-bit hash function $HASH_{256}(x)$ in encryption mode. For simplicity, we assume that $HASH_{256}(x)$ has 4 passes, each of which consists of 32 rounds and also assume that 8 rounds of $HASH_{256}(x)$ is a Markov cipher, which is explained below.

Such an analysis provides a better understanding of the strength of the compression function.

4.3.2 課題抽出

The strategy to perform the differential cryptanalysis can be mainly divided into two parts: In the first part we divide the function into several consecutive sub-functions and try to find differential characteristics with high probability for these sub-functions. In our analysis, each sub-function will consist of several rounds in a certain pass. Hence all rounds in such a sub-function will use the same non-linear Boolean function. In the second part the differential characteristics for each sub-function are concatenated so that they cover the whole cipher. However, it is difficult to do the second part of the analysis when the

characteristics obtained in the first part have complicated forms. For instance, this is the case for SHA-1. In this report, we present a method to solve this difficulty by combining these two parts into a single part.

4.3.3 機能要件

In differential cryptanalysis, two difference operations are often used: $\Delta(X, X') = X \oplus X'$, $\Delta(X, X') = X - X'$. We will consider both cases in our cryptanalysis.

The theoretical background of this method is the theory of Markov ciphers and their connection to differential cryptanalysis introduced by Lai *et al.* in [50]. For an iterated cipher E with the function $Y = T(X, Z)$ which takes the plaintext input as X, the subkey input as Z, we denote the conditional probability that β is the difference $\Delta Y(i)$ of the ciphertext pair after i rounds of S, given that α is the difference of the plaintext pair, by $P(\Delta Y(i) = \beta | \Delta X = \alpha)$.

Recall that a sequence of discrete random variables $v_0, v_1, \ldots, v_r$ is a *Markov chain* if, for $0 \leq i \leq r$,

$$P(v_{i+1} = \beta_{i+1} | v_i = \beta_i, v_{i-1} = \beta_{i-1}, \ldots, v_0 = \beta_0) = P(v_{i+1} = \beta_{i+1} | v_i = \beta_i).$$

A Markov chain is called *homogenous* if $P(v_{i+1} = \beta | v_i = \alpha)$ is independent of i for all α and β. A Markov cipher is defined as follows:

Definition 10 *An iterated cipher with the function T is a Markov cipher if for all choices of α and β,*

$$P(\Delta Y = \beta | \Delta X = \alpha, X = \gamma)$$

is independent of γ when the subkey is uniformly random.

We now state the following theorem using our notation.

Theorem 2 *If an r-round iterated cipher is a Markov cipher and the r round keys are independent and uniformly random, then the sequence of differences $\Delta X = \Delta Y(0), \ldots, \Delta Y(r) = \Delta Y$, is a homogenous Markov chain.*

In the case of $HASH_{256}(x)$, we denote 8 consecutive rounds of E by T. We assume that the cipher E, obtained by iterating T, is a Markov cipher. This allows us to search for differentials rather than characteristics. The goal of our attack is to find a high probability differential for the 4-pass.

4.3.4 モジュール構成

Our goal is to study differential properties of the 4-pass $HASH_{256}(x)$. We will consider low Hamming weight differentials and their propagation: we study the behavior of the 4-pass $HASH_{256}(x)$ compression function when we apply input differentials of weight 1 and 2. At the output, we only observe output differentials of weight 1 and 2. We will check whether these observations are in accordance with the randomness criteria we would expect from a cryptographic hash function.

Let $\mathcal{A}$ be the set of all the bit strings of length 256:

$$\mathcal{A} = \{0, 1\}^{256}.$$

Let $\mathcal{B}$ be the subset of $\mathcal{A}$ where each element has Hamming weight equal to 1:

$$\mathcal{B} = \{\Delta \in \mathcal{A} | Ham(\Delta) = 1\}.$$

Let $\mathcal{C}$ be the subset of $\mathcal{A}$ where each element has Hamming weight equal to 2:

$$\mathcal{C} = \{\Delta \in \mathcal{A} | Ham(\Delta) = 2\}.$$

Let $\mathcal{D}$ be the union set of $\mathcal{B}$ and $\mathcal{C}$:

$$\mathcal{D} = \mathcal{B} \cup \mathcal{C}.$$

Let $\mathcal{E}$ be a set of integers where each element is greater than 0 and is less than or equal to the size of $\mathcal{D}$:

$$\mathcal{E} = \{1, 2, \ldots, 2^8 + \frac{2^8 \cdot (2^8 - 1)}{2}\}.$$

Using the first consecutive 8 rounds in the s-th pass, we build a matrix M_s. To do so, we first define a function g mapping $\mathcal{D}$ to $\mathcal{E}$ in the following manner. If $\Delta \in \mathcal{B}$, let k be the position of 1 in Δ. Otherwise, let h be the high position of 1 in Δ and let l be the low position of 1 in Δ. The function g is defined as follows:

$$g(\Delta) = \begin{cases} k - 1 & \Delta \in \mathcal{B} \\ h - l - 1 + \sum_{i=0}^{l-1} (256 - i) & \Delta \in \mathcal{C}. \end{cases}$$

It is easy to see that g is bijective. The aim of the function g is to establish an ordering for the elements of $\mathcal{D}$.

Now, let's denote the function which consists of the first consecutive 8 rounds in the s-th pass as T_s. To construct a matrix M_s, we randomly choose a (sufficiently large) subset R of $\mathcal{A}$. The cardinality of the subset R is denoted by $\#R = r$. For i and j in $\mathcal{E}$, we define each entry $a_{ij}^{(s)}$ in the matrix M_s as follows:

$$a_{ij}^{(s)} = \frac{\#\{p \in R | g^{-1}(j) = \Delta(T_s(p), T_s(\Delta(p, g^{-1}(i))))\}}{r}.$$

The entry $a_{ij}^{(s)}$ estimates the probability of the differential where the input difference is $g^{-1}(i)$ and the output difference is $g^{-1}(j)$.

4.4 仕様検討

We assume that one pass of $HASH_{256}(x)$ behaves as a 4-round Markov cipher with T_s as the round transformation. Thus the matrix M_s is a transition matrix of the corresponding Markov chain. Raising this matrix to the fourth power as M_s^4 allows us to see the probabilities of 32-round differentials for the s-th pass. Calculating the composition $\mu = \mu_4^4 \circ \mu_3^4 \circ \mu_2^4 \circ \mu_1^4$ allows us to see the differential structure of the 4-pass $HASH_{256}(x)$, where the function μ_s is defined by a matrix multiplication as $\mu_s(X) = X \cdot M_s$. For example, we can see high probability differentials for the whole cipher. What is of most interest now is the highest value in the matrix $M = \mu(I)$. This highest value corresponds to a particular low-weight differential which has a high probability.

4.5 まとめ

汎用評価ツールに関する調査と検討も行い、特に、ハッシュ関数の構成ブロック暗号部の差分攻撃耐性を評価するための方法論を纏めた。

5 ハッシュ関数 MAME の安全性評価

5.1 調査目的

本章では、NIST の公募への提案が想定されるハッシュ関数として MAME を取り上げ、MAME に対する安全性根拠の調査および安全性要件の検討を行う。また、MAME に適用可能な安全性評価手法の調査・検討を行い、評価試行および有効性の評価を行う。

5.2 MAME の仕様

本節では、MAME のアルゴリズムを記述する。

MAME の構造は、大半のハッシュ関数アルゴリズムと同じく 3 つの層からなる。第一層は攪拌処理の主体となるブロック暗号 (の暗号化関数)、第二層は一定長のデータを圧縮する圧縮関数で、圧縮関数は第一層の暗号化関数を内部関数として使用する。そして、第三層は任意長のデータを処理するハッシュ関数で、第二層の圧縮関数の連鎖を定義する。本章ではまず 5.2.1 節で第一層の暗号化関数の仕様を記述する。次に 5.2.2 節で第二層と第三層の構造を記述する。

5.2.1 暗号化関数

この小節では、MAME のベースとなる暗号化関数アルゴリズムを記述する。

概要　ブロック暗号の暗号化関数は固定長の鍵 K とメッセージ M を入力とし、暗号文 C を出力する関数である。暗号化関数 f_E の第一入力を鍵入力、第二入力を平文もしくはメッセージ入力、暗号化関数の出力を暗号文と呼ぶ。

暗号化関数 $f_E(\cdot,\cdot)$ の概要を図 3 に示す。

MAME の第一層となる暗号化関数では、データ処理を 3 つの処理関数に分割しており、図中左から定数生成部、鍵スケジューリング関数、主攪拌関数と呼ぶ。図からもわかるように、いずれも入力に対して単一関数を繰り返し作用させる処理である。3 つの処理関数の処理単位関数をそれぞれラウンド定数生成関数、ラウンド鍵生成関数、ラウンド関数と呼び、それぞれ f_C, f_K, f_R と表す。

定数生成部は定数初期値 $c^{(0)}$ と定数生成関数 f_C からなる。定数生成部はラウンド定数生成関数 f_C の処理ごとにラウンド定数 $C^{(r)}$ を生成する。ラウンド定数 $C^{(r)}$ はラウンド鍵生成関数 f_K への補助入力となる。鍵スケジューリング関数は上記ラウンド鍵生成関数 f_K の繰り返しで構成される関数であり、暗号化関数の入力のうち鍵入力を入力とし、前記定数生成部の出力を補助入力とする。鍵スケジューリング関数はラウンドごとにラウンド鍵 $K^{(r)}$ を生成する。ラウンド鍵 $K^{(r)}$ はラウンド関数 f_R への補助入力となる。主攪拌関数はラウンド関数 f_R の繰り返しからなり、メッセージ入力を入力とするほか、鍵スケジューリング関数の出力を補助入力とし、暗号文を出力する。

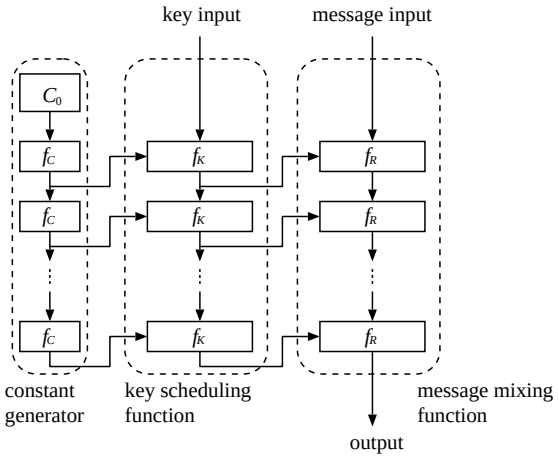

図 3: ハッシュ関数向け暗号化関数の構造

基本パラメータ (入出力長)　ブロック暗号の入力長は、第一入力の長さ (鍵長) が 256 ビット、第二入力の長さ (ブロック長) が 256 ビットである。出力長はブロック長と同じである。

平文を $P = (p_0, p_1, \ldots, p_7)$、暗号文を $E = (e_0, e_1, \ldots, e_7)$ と表せば、主攪拌関数は以下で定義される。

$$\begin{aligned}
(x_0^{(0)}, x_1^{(0)}, \ldots, x_7^{(0)}) &= (p_0, p_1, \ldots, p_7), \\
(x_0^{(r)}, x_1^{(r)}, \ldots, x_7^{(r)}) &= f_R(x_0^{(r-1)}, x_1^{(r-1)}, \ldots, x_7^{(r-1)}), \quad 1 \le r \le 96, \\
(e_0, e_1, \ldots, e_7) &= (x_0^{(96)}, x_1^{(96)}, \ldots, x_7^{(96)}).
\end{aligned}$$

ラウンド関数 f_R は鍵加算、F 関数 F を用いた処理とワード単位の置換からなる。ここで、鍵加算とは鍵スケジューリング関数からの入力 k を x_4 に排他的論理和する処理である。また、F 関数は 2 ワード入力、2 ワード出力の非線形変換である。F 関数の入力はラウンド関数の入力のうち x_4, x_5 であり、その出力は x_6, x_7 に排他的論理和される。F 関数の出力の上位ワードを F_H、下位ワードを F_L と表せば、ラウンド関数 f_R は以下で定義される。

$$\begin{aligned}
x_0^{(r)} &= x_6^{(r-1)} \oplus F(x_4^{(r-1)} \oplus K^{(r)}, x_5^{(r-1)})_H, \\
x_1^{(r)} &= x_7^{(r-1)} \oplus F(x_4^{(r-1)} \oplus K^{(r)}, x_5^{(r-1)})_L, \\
x_2^{(r)} &= x_0^{(r-1)}, \\
x_3^{(r)} &= x_1^{(r-1)}, \\
x_4^{(r)} &= x_2^{(r-1)}, \\
x_5^{(r)} &= x_3^{(r-1)}, \\
x_6^{(r)} &= x_4^{(r-1)}, \\
x_7^{(r)} &= x_5^{(r-1)}.
\end{aligned}$$

次に、F 関数の詳細を記述する。F 関数への入力ワードを a_H, a_L とする。F 関数は S-box 層 $\mathcal{S}$、線形拡散層 $\mathcal{L}$ の 2 層からなる。この 2 層はいずれも 64 ビット入力 64 ビット出力の変換であり、F 関数は 2 つの変換の合成として $F = \mathcal{L} \circ \mathcal{S}$ のように定義される。

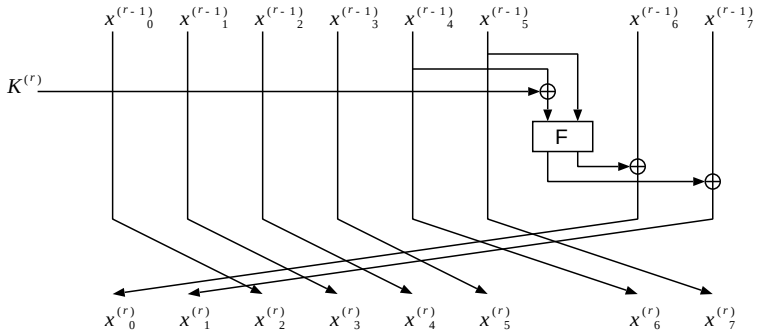

図 4: ラウンド関数 f_R

S-box 層は次で定義される 4 ビット入力、4 ビット出力の置換表 S を用いる。

$$S[16] = \{4, 14, 15, 1, 13, 9, 10, 0, 11, 2, 7, 12, 3, 6, 8, 5\}.$$

S-box S への入力は $a_H||a_L$ の 16 ビットごとの値を集めたものである。すなわち、S-box 層の出力を b_H, b_L と表せば、S-box 層は以下で定義される。

$$b_{H,i+16}||b_{H,i}||b_{L,i+16}||b_{L,i} = S[a_{H,i+16}||a_{H,i}||a_{L,i+16}||a_{L,i}], \quad 0 \le i < 16.$$

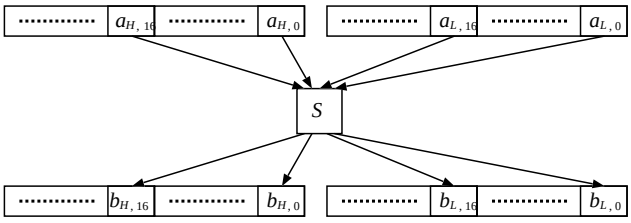

図 5: S-box への入出力の取り方

MAME の S-box 層はビットスライス実装することを前提に設計されている。線形拡散層は巡回シフトと排他的論理和で構成されており、以下で定義される。

鍵スケジューリング関数 ラウンド鍵生成関数 f_K は主攪拌関数のラウンド関数 f_R とほぼ同じ構造を持つが、初期入力は平文ではなく鍵入力であり、鍵加算処理の替わりに定数加算処理を行う。

$$\begin{aligned}
k_0^{(r)} &= k_6^{(r-1)} \oplus F(k_4^{(r-1)} \oplus C^{(r)}, k_5^{(r-1)})_H, \\
k_1^{(r)} &= k_7^{(r-1)} \oplus F(k_4^{(r-1)} \oplus C^{(r)}, k_5^{(r-1)})_L, \\
k_2^{(r)} &= k_0^{(r-1)}, \\
k_3^{(r)} &= k_1^{(r-1)}, \\
k_4^{(r)} &= k_2^{(r-1)}, \\
k_5^{(r)} &= k_3^{(r-1)}, \\
k_6^{(r)} &= k_4^{(r-1)}, \\
k_7^{(r)} &= k_5^{(r-1)}.
\end{aligned}$$

第 r ラウンドのラウンド鍵 $K^{(r)}$ は $k_3^{(r)}$ を使用する。

定数の生成 ラウンド鍵生成関数 f_K に対する補助入力 $C^{(r)}$ は固定の初期値 $c^{(0)}$ から簡単な線形変換 f_C で連続的に生成される。ラウンド定数生成関数は 64 ビット変数 $c^{(r)}$ を用いると次のように表される。

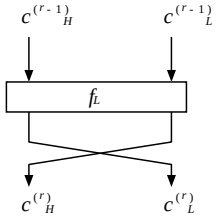

図 6: ラウンド定数生成関数 f_C

$$\begin{aligned}
t_H || t_L &= f_L(c^{(r-1)}), \\
c^{(r)} &= t_L || t_H.
\end{aligned}$$

f_L は線形フィードバックシフトレジスタ (LFSR) の Gaussian 実装である。

定数生成部の初期値 $c^{(0)}$ は次で与えられる。

$$c^{(0)} = \texttt{0xcae1ac3f55054a96}.$$

第 r ラウンドにおけるラウンド定数 $C^{(r)}$ は $c^{(r)}$ の下位 32 ビットを使用する。

5.2.2 ハッシュ関数の構成

ブロック暗号を用いた圧縮関数の構成 MAME では、暗号化関数 f_E を使って以下のように圧縮関数 h を構成する。

$$h(H, M) = f_E(H, M) \oplus M.$$

この構成法は Matyas-Meyer-Oseas の構成法 (MMO) として知られている ([55], 340 ページ参照)。MMO の概略を図 7 に示す。

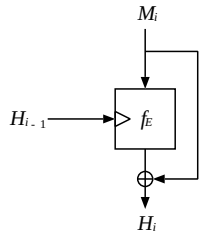

図 7: Matyas-Meyer-Oseas による圧縮関数の構成法

圧縮関数を用いたハッシュ関数の構成 メッセージ M を 256 ビットのメッセージブロックに分割したものを $M_1, M_2, \ldots, M_l$ とする。MAME では MD 構成法に従い以下の手順でメッセージブロックを処理し、最終的に得られる値 H_l をハッシュ値とする。

$$H_i \quad = \quad h(H_{i-1}, M_i).$$

初期値 前記連鎖で用いられる 256 ビットの初期値 $H_0 = (H_{0,0}, H_{0,1}, \ldots, H_{0,7})$ は以下で与えられる。

$$\begin{aligned}
H_{0,0} &= \texttt{0xbc18bf6d}, \\
H_{0,1} &= \texttt{0x369c955b}, \\
H_{0,2} &= \texttt{0xbb271cbc}, \\
H_{0,3} &= \texttt{0xdd66c368}, \\
H_{0,4} &= \texttt{0x356dba5b}, \\
H_{0,5} &= \texttt{0x33c00055}, \\
H_{0,6} &= \texttt{0x50d2320b}, \\
H_{0,7} &= \texttt{0x1c617e21}.
\end{aligned}$$

5.3 MAME の圧縮関数モード 部の安全性評価

To build the compression funtion from a block cipher, MAME uses the MMO mode. The designers of MAME investigate the security of the block cipher to see if they can find some weakness that one can not expect from the PRP.

Hereafter, we study the MMO mode [9]. Although a proof of security for a blockcipher-based hash function in the standard model would be prefered, it has been shown that the PRP assumption is insufficient for building a collision-resistant hash function [93]. Indeed, one can easily imagine a blockcipher $\tilde{E}$:$\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$: that is a good PRP, but which fails when used in the MMO construction. For example, let blockcipher E:$\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$: be a good PRP and consider blockcipher $\tilde{E}$ defined as follows:

表 8: 差分解読法に関する安全性評価

Finilist 暗号	安全性評価
MARS	2^{280} 個以上の選択平文が必要であり、適用困難 ・E 関数 (通常のラウンド関数の F 関数に相当の差分特性確率は概ね 2^{-9} ・16 段の差分特性確率は概ね 2^{-240}
RC6	18 段の最大差分特性確率は概ね 2^{-264} であり、適用不可能
Rijndael	8 段の最大差分特性確率は 2^{-350} であり、適用不可能 ・S-box の最大差分特性確率は 2^{-7}
Serpent	6 段の最大差分特性確率が 2^{-58} 以下 24 段の場合には 2^{-232} 以下となることから、適用困難
Twofish	7 段に対して、2^{131} 個の選択平文が必要であることから、適用困難

$$\tilde{E}(K,X) = \begin{cases} K & X = K \\ E(K,K) & X = E^{-1}(K,K) \\ E(K,X) & otherwise \end{cases}$$

So $\tilde{E}$ is the same blockcipher as E with one change: we now have the invariant that $E(K,K) = K$ for all $K \in \{0,1\}^n$. Clearly $\tilde{E}(K,X)$ is a good PRP since E was: for a randomly chosen key K, $\tilde{E}(K,\cdot)$ is computationally indistinguishable from a random permutation. However, it is trivial to find collisions. Specifically, let H be MMO built on $\tilde{E}$ with $h_0 = 0^n$. Then $H(a \parallel \tilde{E}(0,a) \oplus a) = 0^n$ for 0^n for all $a \in \{0,1\}^n$.

5.4 MAME のブロック暗号部の安全性評価

MAME は、構成要素としてブロック暗号を用いている。このため、ブロック暗号の解読技術や安全性評価手法の適用可能性は高く、特に AES 選定プロセスで進展した技術については、重要であるため、本節では、これらについて広く調査し、その結果を纏める。

5.4.1 AES 選定プロセスにおける finilist の安全性評価

AES 選定プロセスにおける Finilist 暗号に関するアルゴリズムの差分解読法と線形解読法に関する安全性の評価結果を以下に述べる。

5.4.2 AES 会議開催中に提案された解読法

(1) Biham による解読手法 Impossible Differential Cryptanalysis は差分解読法の一種であり、ある特定の差分を有する平文ペアに対し、絶対生成されることがない差分を有する暗号文ペアを利用して鍵の候補

表 9: 線形解読法に関する安全性評価

Finilist 暗号	安全性評価
MARS	2^{128} 個以上の既知平文が必要であり、適用困難 ・4 段の線形特性確率は概ね 2^{-69}
RC6	18 段の RC6 に適用するためには、少なくとも 2^{182} 個の既知平文が必要であり、適用不可能
Rijndael	8 段の最大線形特性確率は 2^{-300} であり、適用不可能 ・S-box の最大線形特性確率は 2^{-6}
Serpent	24 段の最大線形特性確率は 2^{-109} 以下であることから、適用困難
Twofish	12 段に対して、2^{121} 個の既知平文が必要であることから、適用困難

表 10: 提案者らによるその他の攻撃に関する安全性評価

Finilist 暗号	安全性評価
MARS	弱鍵は見つかっていない
RC6	高階差分攻撃や弱鍵の発見に必要なデータを入手することは困難
Rijndael	補間攻撃や関連鍵攻撃に対して安全であるとみられる
Serpent	高階差分攻撃は適用困難 S-box のブール多項式次数が 3 であることから r 段後の出力データの次数は 3r(5 段で 243)
Twofish	S-box は高次の非線形性を有している。 このこと等から、高階差分攻撃、補間攻撃、関連鍵攻撃に対して高い安全性を有する

表 11: 提案者以外による安全性評価結果

Finilist 暗号	安全性評価
MARS	160 ビット鍵では 2^{16} の計算量で等価鍵が見つかる。 128 ビット鍵では 2^{32} の計算量で見つかる。
RC6	256 ビット鍵では、χ 2 検定を用いた攻撃により 15 段に対し適用可能。
Rijndael	128 ビット鍵では、線形和攻撃により、6 段に対し適用可能。
Serpent	S-box の中で、出力データのブール多項式が 2 次になるものが存在。
Twofish	-

を絞り込む方法である。1998 年、Biham らは、31 段の Skipjack(標準 32 段) に対して、2^{41} 個の選択平文・暗号文ペアと 2^{78} 回の暗号化処理が必要とする攻撃を提案した。　Biham は、MARS に対し、そのコアの 7 段の不可能差分を発見し、それに基づき、MARS の 12 段の変形版は解読可能であると見積もった。第 3 回 AES 候補会議 (AES3) においては、Biham は 8 段の不可能差分を発見した。

(2) Rijndael に対しては、Square 攻撃が有効な攻撃法と考えられている。例えば、Rijndael の設計者による解析結果「Square 攻撃により、128 bit 鍵長の場合、標準 10 段のところを 6 段だけ処理したものについては攻撃可能」や、Lucks による解析結果「192 bit の鍵長の Rijndael に対しては、標準 12 段のものが 7 段まで攻撃可能 (第 3 回 AES 候補コンファレンス (AES3))」がある。Lucks の攻撃は、Rijndael 鍵スケジュールの特性を利用したものである。Gilbert と Mininer は 2^{32} 個の選択平文を必要とする 4 段の distinguisher に基づく衝突攻撃を用いて、196 bit と 256 bit の鍵長の 7 段 Rijndael を解読した [29]。

(3) Twofish については、その鍵スケジュール部において二つの性質が発見された [65]。初めは 16 バイトのホワイトニング部分鍵が一様分布でないことである。次は 8 バイト鍵を持つ簡易 Twofish(Feistel 段関数が固定)n に対し、任意の 8 バイト段部分鍵が一様分布でないことである。同じ段部分鍵を与える二つの異なる 8 バイト鍵の例が与えられる。

(4) MARS と RC6 においてはブロック暗号が Davies-Meyer モードで使用されると、その結果のハッシュ関数は使用されている鍵のサイズに直接的に関係する。それゆえ、できる限り長い鍵の使用が望ましい場合も多い。フィンランドの M-J. Saarinen は Davies-Meyer ハッシュモードで長い鍵の使用時における MARS と RC6 の安全性 [85] について研究した。MARS に対しては $2^{\lceil 12 \rceil}$ の計算量で等価な鍵を見つけることができるアルゴリズムを構成した [86]。RC6 に対しては 2^{17} の計算量で擬似等価鍵 (semi equivalent key) を見つけることができることを示した。この結果、ある鍵長を持つ MARS と RC6 で構成された Davies-Meyer ハッシュ関数は安全ではないと考えられる。128,192,256 ビットの鍵長を持つ MARS と RC6 はこの攻撃は適用できない。Saarinen は、これらの暗号の鍵長範囲を制限するべきだと提案している。

5.4.3 AES 会議開催後に提案された解読法

J. Nakahara, I.C. Pavão, "Impossible-Differential Attacks on Large-Block Rijndael," ISC 2007 [66]

ブロック暗号 Rijndael では、ブロック長、鍵長ともに 128, 160, 192, 224, 256 ビットから選択可能である。このうち、ブロック長 128 ビット、鍵長 128, 192, 256 ビットのものが米国標準 AES として採用された。本論文ではラウンドが削減されたブロック長が 160 ビット以上の Rijndael に対する不可能差分 (ID) 攻撃を記述している。暗号化方向に伝播する切り詰め差分及び復号方向に伝播する切り詰め差分から、7 ラウンドの ID 識別者を構成している。ブロック長が大きくなると、拡散にかかるラウンド数が多くなるため、ID 識別者も長くなり、より長いラウンドに対する攻撃が可能となると述べている。

5.5 まとめ

ハッシュ関数 MAME の安全性評価技術について、調査と検討を行なった。MAME の安全性を担う主構成要素はそのブロック暗号部である。ブロック暗号の研究分野は、AES 会議以降、重要な安全性評価手法が広範囲に提案されてきた。本調査では、それらを纏めることにより、MAME のへの適用が可能と考えられる安全性評価手法や脅威を洗い出し、将来の MAME の安全性評価のための指針とした。

参考文献

[1] R.J. Anderson, E. Biham, Tiger: A Fast New Hash Function, In *FSE '96 Proceedings, Lecture Notes in Computer Science 1039,* D. Gollmann, Ed., Springer, pp. 89-97, 1996.

[2] E. Andreeva, G. Neven, B. Preneel, T. Shrimpton, Three-Property Preserving Iterations of Keyless Compression Functions, Ecrypt Hash Workshop 2007,

[3] E. Andreeva, G. Neven, B. Preneel, T. Shrimpton, Seven-Property-Preserving Iterated Hashing: ROX, In *ASIACRYPT '07 Proceedings, Lecture Notes in Computer Science 4833,* pp.130-146, 2007

[4] M. Bellare, P. Rogaway, Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (1993),* pp. 62-73.

[5] D.J. Bernstein, What Output Size Resiste Collisions in a Xor of Independent Expansions?, Ecrypt Hash Workshop 2007,

[6] G. Bertoni, J. Daemen, M. Peeters, G. van Assche, Sponge Functions, Ecrypt Hash Workshop 2007,

[7] E. Biham, A. Shamir, Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.

[8] A. Biryukov, D. Wagner, Advanced slide attacks, In *Eurocrypt '00 Proceedings, Lecture Notes in Computer Science 1807,* B. Preneel, Ed., Springer-Verlag, pp. 589–606, 2000.

[9] J. Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. Cryptology ePrint Archive, Report 2005/210, 2005. `http://eprint.iacr.org/`. Also in *Preproceedings of the 13th Fast Software Encryption Workshop (FSE 2006)*, pages 349–361, 2006.

[10] J. Black, M. Cochran, T. Highland, A Study of the MD5 Attacks: Insights an Improvements, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.262-277, 2006

[11] J. Black, P. Rogaway, and T. Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In *CRYPTO 2002 Proceedings, Lecture Notes in Computer Science 2442*, pages 320-335, 2002.

[12] B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas Jr., C. H. W. Meyer, J. Oseas, S. Pilpel, and M. Schilling. Data authentication using modification detection codes based on a public one-way encryption function, mar 1990. U. S. Patent # 4,908,861.

[13] S. Contini, R. Steinfeld, J. Pieprzyk, K. Matusiewicz, A Critical Look at Cryptographic Hash Function Literature, Ecrypt Hash Workshop 2007,

[14] I. Damgård. Collision free hash functions and public key signature schemes. In *EUROCRYPT '87 Proceedings, Lecture Notes in Computer Science 304*, pages 203–216, 1988.

[15] I. Damgård, A design principle for hash functions, In *Crypto '89 Proceedings, Lecture Notes in Computer Science 435,* G. Brassard, Ed., Springer-Verlag, pp. 416–427, 1990.

[16] M. Daum, Finding Differential Patterns for the Wang Attack, Conference Hash Functions,

[17] B. den Boer, A. Bosselaers, Collisions for the compression function of MD5, In *Eurocrypt '93 Proceedings, Lecture Notes in Computer Science 765,* T. Helleseth, Ed., Springer-Verlag, pp. 293–304, 1993.

[18] C. De Canniere, F. Mendel, C. Rechberger, On the Full Cost of Collision Search fo SHA-1, Ecrypt Hash Workshop 2007,

[19] A. Desai, The security of all-or-nothing encryption: Protecting against exhaustive key search. In *CRYPTO '00 Proceedings, Lecture Notes in Computer Science 1880,* Springer-Verlag, 2000

[20] H. Dobbertin, The status of MD5 after a recent attack, *Cryptobytes,* Vol. 2, No. 2, pp. 1–6, Summer 1996.

[21] O. Dunkelman, B. Preneel, Generalizing th Herding Attack to Concatenated Hashing Schemes, Ecrypt Hash Workshop 2007,

[22] S. Even, Y. Mansour, A construction of a cipher from a single pseudorandom permutation. In *ASIACRYPT '91 Proceedings, Lecture Notes in Computer Science 739,* Springer-Verlag, pp. 210-224, 1991

[23] A. Fiat, A. Shamir, How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO '86 Proceedings, Lecture Notes in Computer Science,* Springer-Verlag, pp. 186-194.

[24] M. Finiasz, P. Gaborit, N. Sendrier, Improved Fast Syndrome Based Cryptographic Hash Function, Ecrypt Hash Workshop 2007,

[25] P. Fouque, G. Leurent, P. Nguyen, Automatic Search of Differential Path in MD4, Ecrypt Hash Workshop 2007,

[26] French Saphir Project (Cryptolog, France Telecom, Ecole Normale Superieure, DCSSI and Gemalto), Revisiting Security Relations between Signature Schemes and Their Inner Hash Functions, Ecrypt Hash Workshop 2007,

[27] P. Gauravaram, W. Millan, and L. May. CRUSH: A new cryptographic hash function using iterated halving technique. In *Proceedings of Cryptographic Algorithms and their Uses 2004*, pages 28–39, 2004.

[28] H. Gilbert, H. Handschuh, Security Analysis of SHA-256 and Sisters, In SAC '03 Proceedings, Lecture Notes in Computer Science 3006, M. Matsui, R. Zuccherato, Eds., Springer-Verlag, pp. 175–193, 2004.

[29] H. Gilbert and M. Minier, *A collision attack on 7 rounds of Rijndael*, Third AES Candidate Conference, 2000.

[30] H. Handschuh, D. Naccache, SHACAL, Submission to the NESSIE project, 2000. Available from `http://www.gemplus.com/ smart/r_d/publications/pdf/HN00shac.pdf`.

[31] S. Hirose. Secure block ciphers are not sufficient for one-way hash functions in the Preneel-Govaerts-Vandewalle model. In *Proceedings of the 9th Selected Areas in Cryptography (SAC 2002), Lecture Notes in Computer Science 2595*, pages 339–352, 2002.

[32] S. Hirose. Provably secure double-block-length hash functions in a black-box model. In *Proceedings of the 7th Internatinal Conference on Information Security and Cryptology (ICISC 2004), Lecture Notes in Computer Science 3506*, pages 330–342, 2005.

[33] S. Hirose. Some plausible constructions of double-block-length hash functions. In *Preproceedings of the 13th Fast Software Encryption Workshop (FSE 2006)*, pages 231–246, 2006.

[34] S. Hirose, J.H. Park, A. Yun, A Simple Variant of the Merkle-Damgård Scheme with a Permutation, In *ASIACRYPT '07 Proceedings, Lecture Notes in Computer Science 4833,* pp.113-129, 2007

[35] J.J. Hoch, A. Shamir, Breaking th ICE - Finding Multicollisions in Iterated Concatenated and Expanded (ICE) Hash Functions, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.179-194, 2006

[36] W. Hohl, X. Lai, T. Meier, and C. Waldvogel. Security of iterated hash functions based on block ciphers. In *CRYPTO '93 Proceedings, Lecture Notes in Computer Science 773*, pages 379–390, 1994.

[37] D. Hong, D. Chang, J. Sung, S. Lee, S. Hong, J. Lee, D. Moon, S. Chee, A New Dedicated 256-Bit Hash Function: FORK-256, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* M.J.B. Robshaw, Ed., Springer, pp. 195-209, 2006.

[38] D. Hong, D. Chang, J. Sung, S. Lee, S. Hong, J. Lee, D. Moon, S. Chee, New FORK-256, Cryptology ePrint Archive 2007/185

[39] S. Indesteege, B. Preneel, Preimages for Reduced-Round Tiger, WEWoRC 2007,

[40] E. Jaulmes, A. Joux, F. Valette, On the security of randomized CBCMAC beyond the birthday paradox limit: A new construction. In *FSE '02 Proceedings, Lecture Notes in Computer Science 2365,* Springer-Verlag, pp. 237-251, 2002

[41] A. Joux, T. Peyrin, Hash Functions and the (Amplified) Boomerang Attack, Ecrypt Hash Workshop 2007,

[42] John Kelsey and Stefan Lucks, Collisions and Near-Collisions for Reduced-Round Tiger, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* M.J.B. Robshaw, Ed., Springer, pp. 111-125, 2006.

[43] J. Kilian, P. Rogaway, How to protect DES against exhaustive key search (An analysis of DESX). Journal of Cryptology 14, 1 (2001), 17-35.

[44] L. Knudsen and F. Muller. Some attacks against a double length hash proposal. In *ASIACRYPT 2005 Proceedings, Lecture Notes in Computer Science 3788*, pages 462–473, 2005.

[45] L. Knudsen and B. Preneel. Hash functions based on block ciphers and quaternary codes. In *ASIACRYPT '96 Proceedings, Lecture Notes in Computer Science 1163*, pages 77–90, 1996.

[46] L. Knudsen and B. Preneel. Fast and secure hashing based on codes. In *CRYPTO '97 Proceedings, Lecture Notes in Computer Science 1294*, pages 485–498, 1997.

[47] L. Knudsen and B. Preneel. Construction of secure and fast hash functions using nonbinary error-correcting codes. *IEEE Transactions on Information Theory*, 48(9):2524–2539, 2002.

[48] L. R. Knudsen, X. Lai, and B. Preneel. Attacks on fast double block length hash functions. *Journal of Cryptology*, 11(1):59–72, 1998.

[49] L. R. Knudsen, C. Rechberger and S.S. Thomsen, Grindahl - a family of hash functions, In *FSE '07 Proceedings, Lecture Notes in Computer Science 4593,* A. Biryukov, Ed., Springer, pp. 39-57, 2007.

[50] X. Lai, J. Massey, Markov Ciphers and Differential Cryptanalysis, In *Eurocrypt '91 Proceedings, Lecture Notes in Computer Science 547,* D. Davies, Ed., Springer-Verlag, pp. 17–38, 1991.

[51] X. Lai and J. L. Massey. Hash function based on block ciphers. In *EUROCRYPT '92 Proceedings, Lecture Notes in Computer Science 658*, pages 55–70, 1993.

[52] S. Lucks. A failure-friendly design principle for hash functions. In *ASIACRYPT 2005 Proceedings, Lecture Notes in Computer Science 3788*, pages 474–494, 2005.

[53] K. Matusiewicz, S. Contini, J. Pieprzyk, Weaknesses of the FORK-256 Compression Function, Cryptology ePrint Archive 2006/317(First version)

[54] K. Matusiewicz, T. Peyrin, O. Billet, S. Contini, J. Pieprzyk, Cryptanalysis of FORK-256, In *FSE '07 Proceedings, Lecture Notes in Computer Science 4593,* A. Biryukov, Ed., Springer, pp. 19-38, 2007.

[55] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[56] K. Matusiewicz, S. Contini, J. Pieprzyk, Weaknesses of the FORK-256 Compression Function, Cryptology ePrint Archive 2006/317(Second version)

[57] F. Mendel, J. Lano, B. Preneel, Cryptanalysis of Reduced Variants of the FORK-256 Hash Function, In *CT-RSA '07 Proceedings, Lecture Notes in Computer Science 4377,* M. Abe, Ed., Springer, pp. 85-100, 2007.

[58] F. Mendel, B. Preneel, V. Rijmen, H. Yoshida, D. Watanabe, Update on Tiger, In *INDOCRYPT '06 Proceedings, Lecture Notes in Computer Science 4329,* R. Barua, T. Lange, Ed., Springer, pp. 63-79, 2006.

[59] Florian Mendel and Vincent Rijmen, Cryptanalysis of the Tiger Hash Function, In *ASIACRYPT '07 Proceedings, Lecture Notes in Computer Science 4833,* K. Kurosawa, Ed., Springer, pp. 536-550, 2007.

[60] F. Mendel, N. Pramstaller, C. Rechberger, V. Rijmen, Analysis of Step-Reduced SHA-256, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.126-143, 2006

[61] F. Mendel, N. Pramstaller, C. Rechberger, V. Rijmen, The Impact of Carries on the Complexity of Collision Attacks on SHA-1, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.278-292, 2006

[62] R. C. Merkle. One way hash functions and DES. In *CRYPTO '89 Proceedings, Lecture Notes in Computer Science 435*, pages 428–446, 1990.

[63] S. Micali, CS-proofs. In *Proceedings of IEEE Foundations of Computing,* pp. 436-453, 1994

[64] I. Mironov, A. Narayanan, Domain Extensions for Random Oracles: Beyond the Birthday-Paradox Bound, Ecrypt Hash Workshop 2007,

[65] F. Mirza and S. Murphy, An observation on the Key Schedule of Twofish, *Second AES Candidate Conference,* 1999.

[66] J. Nakahara, I.C. Pavão, Impossible-Differential Attacks on Large-Block Rijndael, In *Information Security Conference '07 Proceedings, Lecture Notes in Computer Science 4779,* pp.104-117, 2007

[67] M. Nandi. *Design of Iteration on Hash Functions and Its Cryptanalysis.* PhD thesis, Indian Statistical Institute, 2005.

[68] M. Nandi. Towards optimal double-length hash functions. In *Proceedings of the 6th International Conference on Cryptology in India (INDOCRYPT 2005), Lecture Notes in Computer Science 3797*, pages 77–89, 2005.

[69] M. Nandi, W. Lee, K. Sakurai, and S. Lee. Security analysis of a 2/3-rate double length compression function in the black-box model. In *Proceedings of the 12th Fast Software Encryption (FSE 2005), Lecture Notes in Computer Science 35571*, pages 243–254, 2005.

[70] J.B. Nielsen, Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO '02 Proceedings, Lecture Notes in Computer Science 2442,* Springer-Verlag, pp. 111- 126, 2002

[71] National Institute of Standards and Technology, *Secure hash standard,* (FIPS 180-1), 1995

[72] National Institute of Standards and Technology, *Advanced Encryption Standard (AES),* (FIPS 197), 2001

[73] National Institute of Standards and Technology, *Secure Hash Standard (SHS)*, (FIPS 180-2), August 2002.

[74] S. Park, S. H. Sung, S. Chee, J. Lim, On the security of reduced versions of 3-pass HAVAL, In *ACISP '02 Proceedings, Lecture Notes in Computer Science 2384,* J. Seberry, L. Batten, Eds., pp. 406–419, 2002.

[75] T. Peyrin, Cryptanalysis of Grindahl, In *ASIACRYPT '07 Proceedings, Lecture Notes in Computer Science 4833,* K. Kurosawa, Ed., Springer, pp. 551-567, 2007.

[76] N. Pramstaller and V. Rijmen. A collision attack on a double-block-length hash proposal. Cryptology ePrint Archive, Report 2006/116, 2006. `http://eprint.iacr.org/`.

[77] N. Pramstaller, C. Rechberger, V. Rijmen, On Variable Bit-Rotations in SHA-1-like Hash Functions, Conference Hash Functions,

[78] N. Pramstaller, C. Rechberger, V. Rijmen, Breaking a New Hash Function Design Atrategy Called SMASH, Conference Hash Functions,

[79] B. Preneel, R. Govaerts, and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In *CRYPTO '93 Proceedings, Lecture Notes in Computer Science 773*, pages 368–378, 1994.

[80] T. Ristenpart, T. Shrimpton, Building Application-Agile Hash Functions: the MCM Construction, Ecrypt Hash Functions 2007,

[81] T. Ristenpart, T. Shrimpton, How to Build a Hash Function From Any Collision-Resistant Function, In *ASIACRYPT '07 Proceedings, Lecture Notes in Computer Science 4833,* pp.147-163, 2007

[82] R. Rivest, The MD5 message-digest algorithm, *Request for Comments (RFC) 1321,* Internet Activities Board, Internet Privacy Task Force, April 1992.

[83] P. Rogaway, T. Shrimpton, Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance, In *FSE '04 Proceedings, Lecture Notes in Computer Science 3017,* B.K. Roy, W. Meier, Ed., Springer-Verlag, pp. 371-388, 2004.

[84] M. Saarinen, Cryptanalysis of Block Ciphers Based on SHA-1 and MD5, In *FSE '03, Lecture Notes in Computer Science 2887,* T. Johansson, Ed., Springer-Verlag, pp. 36–44, 2003.

[85] M. Saarinen, A note regarding the hash function use of MARS and RC6, SSH Communications Security Ltd.

[86] M. Saarinen, Equivalent keys in MARS, *Third AES Candidate Conference,* 2000.

[87] Markku-Juhani O. Saarinen, A Meet-in-the-Middle Collision Attack Against the New FORK-256, In *INDOCRYPT '07 Proceedings, Lecture Notes in Computer Science 4859,* K. Srinathan, C.P. Rangan, M. Yung, Ed., Springer, pp. 10-17, 2007.

[88] M.O. Saarinen, Linearization Attacks Against Syndrome Based Hashes, In *INDOCRYPT '07 Proceedings, Lecture Notes in Computer Science 4859,* pp.1-9, 2007

[89] M. Schläffer, E. Oswald, Searching for Differential Paths in MD4, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.242-261, 2006

[90] C.P. Schnorr, Efficient signature generation by smart cards. Journal of Cryptology 4, 3, 161-174, 1991

[91] C. Shannon, Communication theory of secrecy systems. Bell Systems Technical Journal 28, 4, 656-715, 1949

[92] T. Shrimpton, M. Stam, Efficient Collision-Resistant Hashing from Fixed-Length Random Oracles, Conference Hash Functions,

[93] D. Simon, Finding collsions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT '98 Proceedings, Lecture Notes in Computer Science,* Springer-Verlag, pp. 334-345, 1998

[94] B. van Rompay, A. Biryukov, B. Preneel, J. Vandewalle, Cryptanalysis of 3-Pass HAVAL, In *Asiacrypt '03 Proceedings, Lecture Notes in Computer Science 2894,* C. Laih, Ed., Springer-Verlag, pp. 228-245, 2003.

[95] R. Winternitz, A secure one-way hash function built from DES. In *Proceedings of the IEEE Symposium on Information Security and Privacy,* IEEE Press, pp. 88-90, 1984

[96] H. Yoshida, A. Biryukov, B. Preneel, Some Applications of the Biham-Chen Attack, Conference Hash Functions,

[97] H. Yu, X. Wang, A. Yun, S. Park, Cryptanalysis of the Full HAVAL with 4 and 5 passes, In *FSE '06 Proceedings, Lecture Notes in Computer Science 4047,* pp.89-110, 2006

[98] Y. Zheng, J. Pieprzyk, J. Seberry, HAVAL – a one-way hashing algorithm with variable length of output, In *Auscrypt '92 Proceedings, Lecture Notes in Computer Science 718,* J. Seberry, Y. Zheng, Eds., Springer-Verlag, pp. 83–104, 1992.

[99] 青木他, 「128 ビットブロック暗号 Camellia アルゴリズム仕様書」, 第 2 版, 2001.

[100] 小田他, 「Pentium 4 における Camellia の高速実装」, SCIS2006 講演予稿集, 2006.