ハッシュ関数の設計理論

内藤 祐介

電気通信大学

2015 年 3 月

ハッシュ関数の設計理論

内藤 祐介

電気通信大学大学院情報理工研究科

博士（工学）の学位申請論文

2015 年 3 月

ハッシュ関数の設計理論

論文主査委員会

主査: 太田　和夫　教授

委員: 西野　哲朗　教授

委員: 崎山　一男　教授

委員: 吉浦　裕　　教授

委員: 安藤　清　　名誉教授

ハッシュ関数の設計理論

内藤 祐介

論文概要

ランダムオラクル証明法 ($\mathcal{RO}$ 証明法) は暗号学的なハッシュ関数を用いた暗号システムの安全性を保証する最も重要な証明方法である．ハッシュ関数を部品とする暗号アルゴリズムを $\mathcal{C}$ と書き, 具体的なハッシュ関数 H^{P} を $\mathcal{C}$ に組み込んだ $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ を暗号システムと呼ぶ. $\mathcal{RO}$ 証明法を用いて暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を証明するとき，ハッシュ関数 H^{P} を理想化したランダムオラクル ($\mathcal{RO}$) を用いる暗号システム $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$-安全性を証明する．一般には，$\mathcal{RO}$ を H^{P} に置き換えた $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性が保証されていない点が $\mathcal{RO}$ 証明法の問題である.

ハッシュ関数 H^{P} は入力長が任意長の関数で, 入出力長固定のプリミティブ P と, P を用いて任意長の入力を規定長の出力に変換する定義域拡張構造 H から構成される．すなわち, H^{P} は H と P から構成されているので, 入力に対して規定長のランダムな値を出力する $\mathcal{RO}$ とは異なる動作をする.

Indiff. 理論 (Indifferentiability 理論) は，P を理想化して, $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$-安全性から $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を保証する理論で,「ハッシュ関数 H^{P} が Indifferentiable From $\mathcal{RO}$ (IFRO) ($\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO}$ と書く)」ならば，SS（シングルステージ）という制約条件はあるものの，任意の SS の安全性 $\mathcal{G}_s$ と任意の $\mathcal{C}$ に対し，「$\mathcal{C}^{\mathcal{RO}}$ が $\mathcal{G}_s$-安全性ならば $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ も $\mathcal{G}_s$-安全性となる ($\mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ と書く)」ことを保証する[*1]. IFRO 安全性は P を理想化して P の構造を取り除く H^{P} に注目する安全性, すなわち, H の構造に対する安全性である.

IFRO 安全が提案されて以降, Sponge 構造や ChopMD 構造など, 多くの IFRO 安全な定義域拡張構造が提案されている．また, 次世代標準ハッシュ関数として採用されることが決まっている SHA-3 は Sponge 構造を採用しており[*2], IFRO 安全性は定義域拡張構造の標準的な安

[*1] また, $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \Leftarrow \forall \mathcal{G}_s, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ となることも分かっている.

[*2] 米国標準の SHA-2 ハッシュ関数族の SHA-512/224 と SHA-512/256 は定義域拡張構造として IFRO 安全な ChopMD 構造を採用している．一方, SHA-2 の SHA-256 と SHA-512 は IFRO 安全とならない MD 構造を採用している.

全性の概念となっている．

MD (Merkle-Damgård) 構造は Indiff. 理論が提案される以前に設計された構造で, SHA-2 ハッシュ関数族である SHA-256 と SHA-512 の定義域拡張構造に採用されているが, $\mathsf{MD}^{\mathsf{P}} \not\sqsubset \mathcal{RO}$ となることが知られている．すなわち，$\exists \mathcal{G}_s, \exists \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \not\approx_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ （$\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}_s$-安全性を満たすが $\mathcal{C}^{\mathsf{MD}^{\mathsf{P}}}$ は $\mathcal{G}_s$-安全性を満たさない）となる．

Indiff. 理論は, 任意の SS の安全性をカバーするものの, 任意の MS（マルチステージ）の安全性 $\mathcal{G}_m$ をカバーしない．すなわち，$\exists \mathcal{G}_m, \exists \mathsf{H}^{\mathsf{P}}$ s.t. $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \wedge \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \not\approx_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$ となる反例が知られている．

一方, ハッシュ関数に求められる実装に関する要件として, ハッシュ関数は高速に計算できることが求められる．今後, SHA-3 は多くの暗号システムで実装されるハッシュ関数であり, Sponge の高速化は SHA-3 を用いる暗号システムの高速化につながるため重要な研究テーマである．

また, ハッシュ関数に求められる実装に関する要件として, ハッシュ関数のプログラムサイズや回路サイズは小さいことが好ましい．プログラムサイズや回路サイズを小さくすることができる定義域拡張構造 H として，ブロック暗号（例えば，AES）をプリミティブとする倍ブロック長定義域拡張構造がある．ブロック暗号とハッシュ関数を両方実装する場合，ブロック暗号を共通化して使うことができるなら，実装サイズを小さくすることができる．既存研究では, IFRO 安全性より弱い安全性である衝突困難性を満たす H は提案されているものの, IFRO 安全な H は提案されていない．

ところで, 新しいハッシュ関数を設計するために, 既存のハッシュ関数の強力な攻撃法の限界点を見極めることも重要である．差分攻撃法はハッシュ関数の強力な攻撃法の 1 つであるが, Message Modification という新しい攻撃技法が 2004 年に SHA-0 と SHA-1 の差分攻撃法に適用されて, ハッシュ関数の安全性の研究が活発になった．今後, この攻撃法の限界点を見極めることは, ハッシュ関数を設計する立場からも重要である．

本論文ではハッシュ関数の設計を目標として, Indiff. 理論に関して次の 5 つの研究課題を設定して研究する．

1. SS の安全性に対する MD の救済.
2. MS の安全性に対する IFRO 安全な H の救済.
3. Sponge の高速化.
4. P としてブロック暗号を用いた IFRO 安全性な H の提案.
5. Message Modification の改良.

研究課題 1 について． $\exists \mathcal{G}_s, \exists \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \not\approx_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ となることが知られているものの，重要な

$\mathcal{G}_s^*$ と, 実用的な $\mathcal{C}_0$ に対して，$\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}} \not\succ_{\mathcal{G}_s^*} \mathcal{C}_0^{\mathcal{RO}}$ となるとは限らない．また，MD の重要性から $\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}}$ が $\mathcal{G}$-安全性を満たすことが望ましい．

本研究では, IFRO 安全性を弱めた安全性「Indifferentiable from Weakened $\mathcal{RO}$（IFWRO）」を考えて, 以下のことを示す.

- $\mathsf{MD}^{\mathsf{P}} \sqsubset \mathcal{WRO}$ となる $\mathcal{WRO}$ を定義する．$\mathcal{WRO}$ が定義できると, Indiff. 理論より, $\forall \mathcal{G}_s, \forall \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{WRO}}$ が保証できる.
- 次に，重要な $\mathcal{G}_s^*$ と実用的な $\mathcal{C}_0$ に対し，個別に $\mathcal{C}_0^{\mathcal{WRO}}$ が $\mathcal{G}_s^*$-安全性を満たすことを示す.
- 以上の 2 点から, $\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}}$ の $\mathcal{G}_s^*$-安全性を保証する.

研究課題 2 について． $\exists \mathcal{G}_m, \exists \mathcal{C}, \exists \mathsf{H}^{\mathsf{P}}$ s.t. $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \wedge \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \not\succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$ となることが知られているものの，重要な $\mathcal{G}_m^*$ と実用的な $\mathcal{C}_0$ に対して，$\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}} \not\succ_{\mathcal{G}_m^*} \mathcal{C}_0^{\mathcal{RO}}$ となることは限らない．また，Sponge と ChopMD の重要性から，$\mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\}$, 重要な $\mathcal{G}_m^*$, 実用的な $\mathcal{C}_0$ に対して，$\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m^*} \mathcal{C}_0^{\mathcal{RO}}$ となっていることが望ましい．

Reset Indifferentiability (Reset Indiff.) 理論は MS の安全性をカバーし，「ハッシュ関数 H^{P} が Reset Indifferentiable from $\mathcal{RO}$ (RIFRO) ($\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{RO}$ と書く)」ならば，「$\forall \mathcal{G}_m, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$ となる」ことを保証する．一方, $\mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\}$ は $\mathsf{H}^{\mathsf{P}} \not\sqsubset_r \mathcal{RO}$ となることが知られている.

本研究では, RIFRO 安全性を弱めた「Reset Indifferentiable from $\mathcal{WRO}$ (Weakened $\mathcal{RO}$)」を考えて, 以下のことを示す.

- $\forall \mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\}$ に対して, $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{WRO}$ となる $\mathcal{WRO}$ を定義する．$\mathcal{WRO}$ が定義できると, Reset Indiff. 理論より, $\forall \mathcal{G}_m, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{WRO}}$ が保証できる.
- 次に, 重要な $\mathcal{G}_m^*$ と実用的な $\mathcal{C}_0$ に対し，個別に $\mathcal{C}_0^{\mathcal{WRO}}$ が $\mathcal{G}_m^*$-安全を満たすことを示す.
- 以上の 2 点を示して, $\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}}$ が $\mathcal{G}_m^*$-安全を満たすことを保証する.

研究課題 3 について． Sponge は入力値を処理する Absorbing ステップと出力値を計算する Squeezing ステップから構成される．Sponge では, Absorbing ステップの内部パラメータは最適な値が設定されているものの，Squeezing ステップのパラメータはチューニングによる高速化の余地が残されている．

本研究では，Sponge の IFRO 安全性について，従来より厳密な解析を行うことで，最適な

安全性証明を行う．最適な IFRO 安全性証明より，Squeezing ステップの最適なパラメータを与えることにより, Sponge の高速化，すなわち，SHA-3 の高速化が可能となることを示す.

研究課題 4 について．本研究では，世界で初めて IFRO 安全な倍ブロック長定義域拡張構造を提案する．提案方式は, 既存方式とほぼ同等の速度と実装サイズを保証し，さらに安全性は衝突困難性から IFRO 安全性に向上した定義域拡張構造である．

研究課題 5 について．本研究では, 既存の SHA-0 に対する差分攻撃法を改良することで, SHA-0 に対する差分攻撃法の限界点を見極める．本研究では, 既存の差分攻撃法で用いられる手法 Message Modification 技法を改良した, Submarine Modification を提案する．次に, Submarine Modification を SHA-0 に適用し, SHA-0 の衝突困難性は, 既存結果で 2^{39} 回の SHA-0 演算で破られていたが, 本研究で 2^{36} 回の SHA-0 演算で破れることを示す.

Design of Cryptographic Hash Functions

Yusuke Naito

Abstract

A Random Oracle methodology ($\mathcal{RO}$ methodology) is one of the important techniques to prove security of hash-based cryptosystems. Hereafter, we denote such system by $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ that consists of a concrete cryptographic algorithm ($\mathcal{C}$) and a concrete hash function (H^{P}). When proving the security of $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ by this methodology, firstly H^{P} is replaced with a $\mathcal{RO}$ which is an ideal hash function, secondly the security of $\mathcal{C}^{\mathcal{RO}}$ is proved. On the other hand, there is a problem of this methodology that the security of $\mathcal{C}^{\mathcal{RO}}$ does *not* imply that of $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$.

A hash function H^{P} is a function that accepts an arbitrary length value and returns the fixed length value. H^{P} consists of two algorithms. One is a domain extension algorithm denoted by H, which defines a structure of a hash function using P. The other is a primitive denoted by P, which is a fixed input and output function. Generically speaking, H^{P} does not behave like a $\mathcal{RO}$, since a $\mathcal{RO}$ is a monolithic function but H^{P} is not.

Indiff. theory (Indifferentiability theory) fill in the security gap between $\mathcal{C}^{\mathcal{RO}}$ and $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ where if "H^{P} is indifferentiable from a $\mathcal{RO}$ denoted by $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO}$" then for any single-stage (SS) security denoted by $\mathcal{G}_s$-security, "$\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ is at least as $\mathcal{G}_s$-secure as $\mathcal{C}^{\mathcal{RO}}$, denoted by $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$"[*3]. We call the security of H^{P} "IFRO security", which is indifferentiable from a $\mathcal{RO}$. Note that when considering IFRO security, the ideal primitive is considered, whereby, IFRO security is of domain extension algorithms.

Indeed, many IFRO secure hash functions have been designed. The typical hash functions are the hash functions of the SHA-3 family and of the SHA-2 family. SHA-3 makes use of the Sponge construction and SHA-2 makes use of the ChopMD construction. Thus, IFRO security is the most important notion of domain extension algorithms.

[*3] Specifically, it was proven that $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \Leftrightarrow \forall \mathcal{G}_s, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$.

The MD (Merkle-Damgård) construction is a domain extension algorithm that was proposed before IFRO security was introduced. It is employed as the domain extension algorithms of SHA-256 and SHA-512 of the SHA-2 family, it was proven that $\mathsf{MD}^{\mathsf{P}} \not\sqsubset \mathcal{RO}$ even if P is ideal. This implies that $\exists \mathcal{G}_s, \exists \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \not\approx_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ ($\mathcal{C}^{\mathcal{RO}}$ is $\mathcal{G}_s$-secure but $\mathcal{C}^{\mathsf{MD}^{\mathsf{P}}}$ is not).

Indiff. theory covers any SS security but does not cover any multi-stage (MS) security denoted by $\mathcal{G}_m$-security. Specifically, it was proven that $\exists \mathcal{G}_m, \exists \mathsf{H}^{\mathsf{P}}$ s.t. $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \wedge \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \not\approx_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$.

A general requirement to a hash function is that the time spent of a hash function should be short as much as possible. Especially, it is desirable that standard hash functions meet this requirement. Hence it is important to improve the time spent of Sponge, which is employed in SHA-3.

Another requirement is that the program/circuit size should be small as much as possible. A double-block-length (DBL) construction, which is a domain extension algorithm, offers hash functions satisfied with the requirement. The constructions are constructed from a blockcipher such as AES. When a blockcipher and a hash function are implemented at the same time, one can reduce the program/circuit size. Indeed, many DBL construction with collision security have been proposed but a DBL construction with IFRO security has not.

It is important to evaluate the limitation of powerful attacks on hash functions in order to design a new hash function. The differentiable attack is one of such attacks which has broken many important hash functions. The message modification technique is a key technique of the attack and due to this technique collision resistance of the important hash functions such as SHA-0 and SHA-1 was broken. Thus evaluating the limitation of this technique is the important research topic.

The goal of this paper is to establish a design methodology of hash functions. Hence this paper tackles the following five research topics.

1. Salvaging the MD construction for SS security.
2. Salvaging IFRO secure constructions for MS security.
3. Improving the time spent of Sponge.
4. Designing a DBL construction which is IFRO secure.
5. Improving the message modification technique.

Our result for the topic 1. Though it was proven that $\exists \mathcal{G}_s, \exists \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \not\succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$, it has not been proven for important SS security (denoted by $\mathcal{G}_s^*$) and practical cryptographic algorithms (denoted by $\mathcal{C}_0$). Due to the importance of the MD construction, it is desirable to be satisfied that $\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}} \succ_{\mathcal{G}_s^*} \mathcal{C}_0^{\mathcal{RO}}$.

For this motivation, we define a weaker security notion than IFRO, which is "Indifferentiable from Weakened $\mathcal{RO}$ (IFWRO)", then prove the $\mathcal{G}_s^*$-security of $\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}}$ by the following steps.

- Define $\mathcal{WRO}$ so that $\mathsf{MD}^{\mathsf{P}} \sqsubset \mathcal{WRO}$. Then Indiff. theory guarantees that $\forall \mathcal{G}_s, \forall \mathcal{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{WRO}}$.
- Prove that $\mathcal{C}_0^{\mathcal{WRO}}$ is $\mathcal{G}_s^*$-secure.
- Conclude that $\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}}$ is $\mathcal{G}_s^*$-secure.

Result for the topic 2. Though it was proven that $\exists \mathcal{G}_m, \exists \mathcal{C}, \exists \mathsf{H}^{\mathsf{P}}$ s.t. $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \wedge \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \not\succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$, it has not been proven for important MS security (denoted by $\mathcal{G}_m^*$) and practical cryptographic algorithms (denoted by $\mathcal{C}_0$). Due to the importance of Sponge and ChopMD, it is desirable to be satisfied that $\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m^*} \mathcal{C}_0^{\mathcal{RO}}$ for $\mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\}$.

Reset indiff. theory (reset indifferentiability theory) covers all MS security notions where "if H^{P} is reset indifferentiable from a $\mathcal{RO}$ (RIFRO) (denoted by $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{RO}$)" then "$\forall \mathcal{G}_m, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$". On the other hand it was proven that $\mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\} : \mathsf{H}^{\mathsf{P}} \not\sqsubset_r \mathcal{RO}$.

So we define a weaker security notion than RIFRO which is "Reset Indifferentiable from $\mathcal{WRO}$ (Weakened $\mathcal{RO}$)", then prove the $\mathcal{G}_m^*$-security of $\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}}$ by the following steps.

- Define $\mathcal{WRO}$ such that $\forall \mathsf{H} \in \{\mathsf{Sponge}, \mathsf{ChopMD}\} : \mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{WRO}$. Then reset indiff. theory ensures that $\forall \mathcal{G}_m, \forall \mathcal{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{WRO}}$.
- Prove that $\mathcal{C}_0^{\mathcal{WRO}}$ is $\mathcal{G}_m^*$-secure.
- Conclude that $\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}}$ is $\mathcal{G}_m^*$-secure.

Result for the topic 3. Sponge consists of the absorbing step and the squeezing step. The absorbing step is to operate an input value on the hash calculation and the squeezing step is to produce the output value.

So far, the optimal parameter of Sponge has not been shown. Hence we first gives the optimal proof of Sponge. Then we show the optimal parameter of the squeezing step by using the proof. Consequently, the time spent of Sponge can be improved.

Result for the topic 4. In this paper, we propose a first time DBL construction which is IFRO secure. Moreover, our construction achieves the same level of speed as the existing DBL constructions that are collision resistant[*4].

Result for the topic 5. In this paper, we evaluate the limitation of the differentiable attack on SHA-0. First, we propose a new message modification called submarine modification. Secondly, we show that this modification improves the complexity of finding a collision of SHA-0 from 2^{39} SHA-0 evaluations to 2^{36} SHA-0 evaluations.

[*4] Collision security is weaker than IFRO security.

目次

記号一覧

- $||$: ビット結合. 例えば, $011||100 = 011100$.
- $x \leftarrow y$: 値 y を変数 x に代入する.
- $x \xleftarrow{\$} X$: 集合 X から 1 つ要素をランダムに選んで変数 x に代入する.
- $\oplus$: 排他的論理和.
- $|x|$: ビット列 x のビット長
- $x[i,j]$: b bit の値 x の i bit 目から j bit 目の値. 例えば, $x = 01101001$ の場合, $x[3,5] = 101$ となる.
- $\emptyset$: 空集合
- ε: 空列
- $A \xleftarrow{\cup} x$: 集合 A, ある値 x に対して, $A \cup \{x\}$ を A に代入する.
- $A \xleftarrow{\cup} C$: 集合 A, C に対して, $A \cup C$ を A に代入する.
- $div(r, m)$: $l \times r$ bit の値 m を r bit の値 $m_1, \ldots, m_l$ に分割して, これらの値を出力する関数. $m_1 \| \cdots \| m_l = m$ である.
- $\mathcal{A}^{\mathcal{O}} \Rightarrow y$: アルゴリズム $\mathcal{A}$ (例えば, 攻撃者や識別者) がオラクル $\mathcal{O}$ と対話して y を出力するイベント.
- $\exists_1 M$ s.t. $F(M) = \mathsf{true}$: ブール関数 F に対して, $F(m) = \mathsf{true}$ となる F の入力値 m が 1 つだけ存在するイベント.
- ベクトルをボールド体で書くことにする. 例えば, $\mathbf{x}$.
- $|\mathbf{x}|$: ベクトル $\mathbf{x}$ のベクトル長.
- $\mathbf{x}[i]$: i 番目のベクトルとする. $1 \leq i \leq |\mathbf{x}|$ である.
- $bit_j(\mathbf{x})$: $\mathbf{x}[1] \| \ldots \| \mathbf{x}[|\mathbf{x}|]$ の左から j 番目のビット.
- $\mathsf{Time}(A)$: アルゴリズム A の最大動作ステップ数.

第 1 章

序論

1.1 研究背景

情報セキュリティは, 例えばインターネットのような, 不特定多数の利用者が接続可能なネットワーク環境などで，安全に情報をやりとりするするために欠かせない技術である. 暗号技術は, 情報セキュリティ技術の 1 つで, 例えば, 暗号化, デジタル署名, メッセージ認証などから構成される.

これらの暗号アルゴリズムは暗号学的なハッシュ関数を部品として用いる. 暗号アルゴリズム $\mathcal{C}$ に具体的な暗号学的なハッシュ関数 H^{P} を組み込んだ $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ を暗号システムと呼ぶ.

1.1.1 暗号学的なハッシュ関数

暗号学的なハッシュ関数は, 安全な暗号アルゴリズム $\mathcal{C}$ を設計するための $\mathcal{C}$ の部品として用いられる. 以降,「暗号学的な」を省略する.

ハッシュ関数は任意長の値を入力として受理し, 固定長または任意長の値を出力する関数である. 固定長の値を出力する代表的なハッシュ関数として米国標準ハッシュ関数である SHA-2 ハッシュ関数族がある [46]. SHA-2 ハッシュ関数族は SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 で, 出力長はそれぞれ 224bit, 256bit, 384bit, 512bit, 224bit, 256bit である. また, 次世代の米国標準ハッシュ関数 SHA-3 に選ばれた Keccak ハッシュ関数族は, 出力長として, 256, 384, 512, そして任意長をサポートする [13].

ハッシュ関数の設計では, 以下の 2 ステップに分けた設計方法が用いられる.

1. 入出力長が固定の内側の部品であるプリミティブ P を設計する.
2. プリミティブ P を用いてハッシュ関数の構造を定義する定義域拡張構造 H を設計する.

ここで, プリミティブ P と定義域拡張構造 H を組み合わせて設計したハッシュ関数を H^{P} と書く.

ハッシュ関数の安全性で考えるべき安全性は, 代表的なものとして, IFRO 安全性 (Indifferentiability from a Random Oracle 安全性), 衝突困難性, 一方向性などがある. これらは, 達成することが難しい順番に並んでいる. すなわち, IFRO 安全なハッシュ関数性は衝突困難性, 一方向性などの安全性も満たすことを意味する. よって, ハッシュ関数の設計では, IFRO 安全性を満たすハッシュ関数を設計することが期待される.

1.1.2 $\mathcal{RO}$ 証明法

$\mathcal{RO}$ 証明法（ランダムオラクル証明法）はハッシュ関数 H^{P} を用いた暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を保証する最も重要な証明技法である [7]. 暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を証明する場合，ハッシュ関数 H^{P} を理想化した $\mathcal{RO}$（ランダムオラクル）を用いる暗号システム $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$-安全性を証明する.

$\mathcal{RO}$ 証明法を用いた暗号アルゴリズムの設計の利点は安全性証明が容易になる点であり，$\mathcal{RO}$ 証明法は効率的な暗号アルゴリズムの設計や高機能な暗号アルゴリズムの設計をする際に多く用いられる [7]. 例えば米国標準暗号となっている重要な暗号アルゴリズム PSS 署名 [9], OAEP 暗号 [8], RSA-KEM [52] は $\mathcal{RO}$ 証明法を用いて設計されている.

しかし，$\mathcal{RO}$ は構造を持たない一枚岩のランダム関数であるのに対し，H^{P} は H の構造と P の構造を持つ関数であるため，$\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$-安全性は $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を保証しない点が $\mathcal{RO}$ 証明法の課題である.

1.1.3 $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の証明

Coron らは, P を理想化して, $\mathcal{C}$ に H を組み込んだ $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を証明する方法を研究した [24]. この安全性を証明する 1 つの方法は，$\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性をダイレクトに証明する方法である. しかし，この方法は，$\mathcal{C}$ と H ごとに $\mathcal{G}$-安全性の証明を与える必要があり手間がかかる. また，$\mathcal{G}$-安全性の証明で $\mathcal{C}$ の構造に加え H の構造を考える必要があるため，証明が複雑になる.

1.1.4 Indifferentiability from a $\mathcal{RO}$

Coron らは，ダイレクト証明の問題を解決するために，2004 年に Maurer らによって提案された Indiff. 理論（Indifferentiability） [40] をハッシュ関数の設計に適用した [24].

Indiff. 理論では, 2 つの関数 F, G に対して, 「F is indifferentiable from G（$F \sqsubset G$ と書く)」ならば, 安全性として SS（シングルステージ）という制約がつくものの，任意のシングルステージゲームで定義される安全性 $\mathcal{G}_s$ と任意の暗号アルゴリズム $\mathcal{C}$ に対して，「$\mathcal{C}^G$ が $\mathcal{G}_s$-安全性を満たすならば $\mathcal{C}^F$ も $\mathcal{G}_s$-安全性を満たす ($\mathcal{C}^F \succ_{\mathcal{G}_s} \mathcal{C}^G$ と書く)」ことを保証する.

すなわち，$F = \mathsf{H}^{\mathsf{P}}$，$G = \mathcal{RO}$ とすると，Indiff. 理論から以下のことを保証することができる．ここで, シングルステージゲームの安全性の集合を $\mathbf{G}_s$, 暗号アルゴリズムの集合を $\mathbf{C}$ と書く.

$$\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \Rightarrow \forall \mathcal{G}_s \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$$

すなわち，Indifferentiable from a $\mathcal{RO}$（IFRO）安全なハッシュ関数 H^{P} を用いれば，$\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}_s$-安全性から，$\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}_s$-安全性を保証できる[*1].

IFRO 安全性は, P を理想化した H^{P} を考えるため, 定義域拡張構造 H に対する安全性であることに注意されたい.

1.1.5 IFRO 安全なハッシュ関数

Coron らによって，IFRO 安全性が考えられて以降, Sponge [11] や ChopMD [24] など多くの IFRO 安全な定義域拡張構造が提案されている．例えば, Sponge は次世代米国標準ハッシュ関数 SHA-3 の定義域拡張構造に採用されている [13]．また，米国標準ハッシュ関数である SHA-2 ハッシュ関数族に, 2012 年まで IFRO 安全となる定義域拡張構造を用いるハッシュ関数は含まれていなかったが，2012 年に IFRO 安全性を満たす ChopMD [24] を用いる SHA-512/224，SHA-512/256 が SHA-2 ハッシュ関数族に追加された [46].

この標準化動向からわかるように，IFRO 安全性は定義域拡張構造の最も重要な安全性であり，現在，ハッシュ関数の設計において標準的な安全性となっている.

1.2 研究課題

本研究では IFRO 安全性に関する課題として, 以下の 5 つの課題を設定する.

[*1] $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \Leftarrow \forall \mathcal{G}_s \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ であることも示せている.

1.2.1 課題 1

Merkle-Damgård 構造 (MD) は Indiff. 理論が提案される以前に設計された定義域拡張構造で [25, 41], SHA-2 ハッシュ関数族に属する SHA-256 と SHA-512 の定義域拡張構造に採用されている [46] が, $\mathsf{MD}^{\mathsf{P}} \not\sqsubseteq \mathcal{RO}$ となることが知られている [24]. すなわち,「$\exists \mathcal{G}_s \in \mathbf{G}_s, \exists \mathcal{C} \in \mathbf{C}$: $\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}_s$-安全性を満たすが $\mathcal{C}^{\mathsf{MD}^{\mathsf{P}}}$ は $\mathcal{G}_s$-安全性を満たさない ($\mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \not\preceq \mathcal{C}^{\mathcal{RO}}$ と書く)」となる．一方で，MD の重要性から，重要な $\mathcal{G}_s^* \in \mathbf{G}_s$ と実用的な $\mathcal{C}_0 \in \mathbf{C}$ に対して，$\mathcal{C}_0^{\mathsf{MD}^{\mathsf{P}}}$ が $\mathcal{G}_s^*$-安全性を満たすことが望ましい.

1.2.2 課題 2

近年，複数の攻撃者が結託することができる強力な攻撃を想定した MS（マルチステージ）で定義される安全性が考えられている [4]. MS の安全性を考えると，検索可能暗号 [3], ヘッジ暗号 [4]，重複除外暗号 [6] などの SS で実現できなかった重要な暗号システムの安全性を定義できる．また，多くの実用的な暗号アルゴリズム $\mathcal{C}$ は, $\mathcal{C}^{\mathcal{RO}}$ が MS の安全性 $\mathcal{G}_m$ を満たすように, 設計されている．しかし，Indiff. 理論は MS の安全性をカバーせず，MS の安全性の集合を $\mathbf{G}_m$, IFRO 安全な H の集合を $\mathbf{H}$ と書くと, $\exists \mathcal{G}_m \in \mathbf{G}_m, \exists \mathsf{H} \in \mathbf{H}$ s.t. $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \not\preceq_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$ となることが知られている．一方で，IFRO 安全の重要性から，重要な $\mathcal{G}_m^* \in \mathbf{G}_m$ と実用的な $\mathcal{C}_0 \in \mathbf{C}$ と重要な $\mathsf{H} \in \mathbf{H}$ に対して，$\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}}$ が $\mathcal{G}_m^*$-安全性を満たすことが望ましい.

1.2.3 課題 3

ハッシュ関数に求められる実装に関する要件として，ハッシュ関数は高速に計算できることが求められる．次世代標準ハッシュ関数 SHA-3 は，今後多くの暗号システムで実装されるハッシュ関数であり，SHA-3 の高速化，すなわち Sponge の高速化は SHA-3 を用いる暗号システムの高速化につながる.

Sponge は入力値を処理する Absorbing ステップと出力値を計算する Squeezing ステップから構成される．Absorbing ステップの内部パラメータは最適な値となっていることが知られているものの，Squeezing ステップの内部パラメータの最適性は示されておらず，パラメータのチューニングによる Sponge の高速化が見込める.

1.2.4 課題 4

ハッシュ関数に求められる実装に関する要件として, ハッシュ関数のプログラムサイズや回路サイズは小さいことが好ましい. プログラムサイズや回路サイズを小さくすることができる定義域拡張構造として，ブロック暗号（例えば，AES）をプリミティブとする倍ブロック長定義域拡張構造がある [20, 42, 34, 37, 47, 38, 39]. ブロック暗号とハッシュ関数を両方実装する場合，ブロック暗号を共通化して使うことができるなら，実装サイズを小さくすることができる. これまで, IFRO 安全性より弱い安全性である衝突困難性を満たす倍ブロック長定義域拡張構造 [34, 37, 47] は提案されているものの, IFRO 安全な倍ブロック長定義域拡張構造は提案されていない.

1.2.5 課題 5

新しいハッシュ関数を設計するために, 既存のハッシュ関数の強力な攻撃法の限界点を見極めることも重要である. ハッシュ関数の強力な攻撃法の 1 つに差分攻撃法 [62, 60] があるが, この攻撃法は新しい攻撃法であり, この攻撃法の限界点が見極められていない.

1.3 研究成果

本研究は上記の 5 つの課題に対して以下の答えを与える.

1.3.1 課題 1 の研究成果

本研究では, まず, Weakened $\mathcal{RO}$ 証明法 ($\mathcal{WRO}$ 証明法) を提案する.

1. $\mathsf{MD}^{\mathsf{P}} \sqsubset \mathcal{WRO}$ となるように $\mathcal{RO}$ を弱めた $\mathcal{WRO}$ を定義する. これが証明できると, Indiff. 理論から, $\forall \mathcal{G}_s \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{MD}^{\mathsf{P}}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{WRO}}$ を保証できる.
2. ある重要な安全性ゲーム $\mathcal{G}_s^* \in \mathbf{G}_s$ とある実用的な暗号アルゴリズム $\mathcal{C}_0 \in \mathbf{C}$ に対して, $\mathcal{C}_0^{\mathcal{WRO}}$ が $\mathcal{G}_s^*$-安全性を満たすことを個別に証明する.
3. 以上の 2 点を示して, $\mathcal{C}_0^{\mathsf{MD}}$ が $\mathcal{G}_s^*$-安全性を満たすことを保証する.

次に, $\mathcal{WRO}$ 証明法を用いて, 実用的な暗号アルゴリズムである FDH 署名, PSS 署名, OAEP 暗号, RSA-KEM が MD を用いた時に安全となることを証明する.

MD を定義域拡張構造として用いた SHA-256 と SHA-512 を PSS 署名, OAEP 暗号, RSA-KEM に組み込む暗号システムは広く実装されているため, 現在運用されている暗号システムの安全性を保証する重要な成果となる.

本研究の詳細は 3 章で述べる. なお, 本研究は本論文に関連した研究業績の「査読付き論文誌 1」と「査読付き国際会議 5」をまとめたものである.

1.3.2 課題 2 の研究成果

本研究では, Ristenpart らのマルチステージ (MS) の安全性をカバーする Reset Indiff. 理論 [51] (Reset Indifferentiability) を利用する. Reset Indiff. 理論は，2 つの関数 F, G に対して，もし，「F is reset indifferentiable from G ($F \sqsubset_r G$ と書く)」ならば「$\forall \mathcal{G}_m \in \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^F \succ_{\mathcal{G}_m} \mathcal{C}^G$」を保証する.

ここで, $F = \mathsf{H}^{\mathsf{P}}$, $G = \mathcal{RO}$ とすると, 以下のことが成り立つ.

$$\forall \mathcal{G}_m \in \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{RO} \Rightarrow \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$$

一方, Sponge や ChopMD などの IFRO 安全な定義域拡張構造 H に対し, 以下のことが示されている [51].

$$\exists \mathcal{G}_m \in \mathbf{G}_m, \exists \mathcal{C} \in \mathbf{C} : \mathsf{H}^{\mathsf{P}} \not\sqsubset_r \mathcal{RO}$$

そこで, 本研究では, 課題 1 の研究成果の Indiff. 理論を用いた $\mathcal{WRO}$ 証明法を Reset Indiff. 理論に拡張する. Reset Indiff. 理論を用いた $\mathcal{WRO}$ 証明法の手順は以下の通りである.

1. 重要な $\mathsf{H} \in \mathbf{H}$ に対して, $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{WRO}$ となる $\mathcal{RO}$ を弱めた $\mathcal{WRO}$ を定義する. $\mathcal{WRO}$ が定義できると, Reset Indiff. 理論より $\forall \mathcal{G}_m \in \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{WRO}}$ を保証できる.
2. ある重要な $\mathcal{G}^*_m \in \mathbf{G}_m$ とある実用的な $\mathcal{C}_0 \in \mathbf{C}$ に対して, $\mathcal{C}_0^{\mathcal{WRO}}$ が $\mathcal{G}^*_m$-安全性を満たすことを証明する.
3. 以上の 2 点から, $\mathcal{C}_0^{\mathsf{H}^{\mathsf{P}}}$ が $\mathcal{G}^*_m$-安全性を満たすことを保証する.

次に, $\mathcal{WRO}$ 証明法を用いて, $\mathcal{RO}$ を用いた時に CDA 安全 [3, 4] となる実用的な暗号アルゴリズム EwH 暗号と REwH 暗号 [3, 4] に対して, Sponge や ChopMD を用いた暗号システムが CDA 安全となることを証明する.

現在，MS の安全性を満たす暗号アルゴリズムの設計は研究段階である. しかし, この研究は多くの研究者に注目されている分野で，今後，この暗号システムが実用化され広く実装され

る可能性がある．本研究は，MS の安全性が求められる暗号システムの実用化に向けてプラスとなる結果を与えた重要な結果である．

本研究の詳細は 4 章で述べる．なお, 本研究は本論文に関連した研究業績の「査読付き国際会議 2」をまとめたものである．

1.3.3 課題 3 の研究成果

本研究では，Sponge の IFRO 安全性を厳密に証明し, Sponge の最適な IFRO の安全性証明を与える．そして, この証明をもとに Squeezing ステップをチューニングし, Squeezing ステップの最適なパラメータの値を示す．この最適な値を用いることで，Sponge の高速化, すなわち Sponge を用いた SHA-3 ハッシュ関数族が安全性を担保したまま高速化できるため, SHA-3 の仕様に影響を与える可能性のある重要な研究成果である．

本研究の詳細は 5 章で述べる．なお, 本研究は本論文に関連した研究業績の「査読付き国際会議 1」をまとめたものである．

1.3.4 課題 4 の研究成果

本研究では，既存の倍ブロック長定義域拡張構造 [34, 37, 47] に 2 回のブロック暗号の計算を加えた方式を提案し，提案方式が IFRO 安全となることを示す．

提案方式は世界で初めての IFRO 安全となる倍ブロック長構造である．提案方式の計算量は既存方式と比べ 2 回のブロック暗号の計算が余分にかかるものの, この 2 回のブロック暗号の演算は既存ハッシュ関数の計算に比べて無視できる計算量である．提案方式の実装サイズは既存の倍ブロック長ハッシュ関数と同じである．このように，提案方式は実装性能を保ったまま安全性が向上しており，小型化を目指した製品に対して安全性も保証することができる重要な結果である．

本研究の詳細は 6 章で述べる．なお, 本研究は本論文に関連した研究業績の「査読付き国際会議 3 と 4」をまとめたものである．

1.3.5 課題 5 の研究成果

本研究では, [62] で提案されている SHA-0 に対する差分攻撃法を改良することで, SHA-0 に対する差分攻撃法の限界点を見極める．限界点見極めのために, [62] の Message Modification 技法を改良した, Submarine Modification を提案する．次に, Submarine Modification を使うことで, SHA-0 の衝突困難性を破る計算量を 2^{39} 回の SHA-0 演算から 2^{36} 回の SHA-0 演

算に改良できることを示す.

この結果は, SHA-0 に対する差分攻撃を改良した結果であり, 差分攻撃の限界点を見極める第 1 歩となる結果である.

本研究の詳細は 7 章で述べる. なお, 本研究は本論文に関連した研究業績の「査読付き論文誌 2」と「査読付き国際会議 6」をまとめたものである.

1.4 本論文の構成

本論文は, 8 章で構成されている.

1 章は本章であり, 本研究背景, 研究課題, そして課題に対する本研究成果の概要を示した. 2 章では, 本研究で用いる IFRO 安全性を定義し, MD 構造, Sponge 構造, ChopMD 構造を説明し, IFRO 安全性に係わる研究課題について詳しく説明する. 3 章では, SS ゲームにおいて MD 構造を $\mathcal{WRO}$ 証明法を用いて救済する. 4 章では, MS ゲームにおいて IFRO 安全な構造を $\mathcal{WRO}$ 証明法を用いて救済する. 5 章では, Sponge 構造を高速化することを目的として, Sponge 構造の IFRO 安全性を担保できる最適なパラメータを示す. 6 章では, IFRO 安全な倍ブロック長構造を提案する. 7 章では, SHA-0 に対する差分攻撃法の改良方法を提案する. 8 章では, 本研究の成果をまとめ, 結論を述べる.

第 2 章

準備

ハッシュ関数は任意長のデータを固定長のデータに圧縮する関数のことである. ハッシュ関数は公開鍵暗号, デジタル署名, メッセージ認証など多くの暗号システムの部品として使われている重要な暗号アルゴリズムである. ハッシュ関数の安全性の要件は, 一方向性, 第二原像計算困難性, 衝突困難性, そして, 本論文で扱う IFRO 安全性がある. 諸概念を定義した後, 主に IFRO 安全性と関連した既存の研究成果を紹介する.

2.1 はじめに

暗号学的なハッシュ関数は一方向性, 第二原像計算困難性, 衝突困難性の安全性を備えた関数で, 多くの暗号システムの部品として使われる.

ランダムオラクル ($\mathcal{RO}$) 証明法は, 暗号アルゴリズム $\mathcal{C}$ に暗号学的なハッシュ関数 H^{P} を用いた暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性を証明する方法である. $\mathcal{RO}$ は初出のクエリに対して出力をランダムに選んだ値を返し, 繰り返し行われたクエリに対しては過去に出力した値と同じ値を返す理想的なハッシュ関数である. そして, $\mathcal{RO}$ 証明法は $\mathcal{RO}$ を用いた暗号システム $\mathcal{C}^{\mathcal{RO}}$ の安全性を証明することで, $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性を保証する証明法である [7].

しかし, $\mathcal{RO}$ を用いた暗号システム $\mathcal{C}^{\mathcal{RO}}$ は安全だが, 一方向性, 第二原像計算困難性, 衝突困難性を満たすハッシュ関数 H^{P} を用いた暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ は安全では無い暗号アルゴリズム $\mathcal{C}$ が存在する [21]. すなわち, 一方向性, 第二原像計算困難性, 衝突困難性は $\mathcal{RO}$ から H^{P} への置き換えを保証しない.

Indifferentiability の理論 [40] は, 2 つの暗号部品 $\mathcal{V}$ と $\mathcal{U}$ に対して, "$\mathcal{U}$ is indifferentiable from $\mathcal{V}$" が成り立ちたつならば $\mathcal{V}$ から $\mathcal{U}$ への置き換えを保証した理論である. そして, $\mathcal{V} = \mathcal{RO}$, $\mathcal{U} = \mathsf{H}^{\mathsf{P}}$ としたハッシュ関数 H^{P} に対する安全性である IFRO (Indifferentiable

from $\mathcal{RO}$) の安全性を考えることで, $\mathcal{RO}$ から H^{P} への置き換えが保証できる. すなわち, 任意の暗号アルゴリズム $\mathcal{C}$ に対して, H^{P} が IFRO 安全かつ $\mathcal{C}^{\mathcal{RO}}$ が安全ならば $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ も安全となる.

しかし, IFRO 安全性はシングルステージゲームの安全性をカバーするものの, マルチステージゲームの安全性をカバーしないことが示されている [51]. シングルステージの安全性は 1 人の攻撃者に対して安全性を評価するゲームで定義された安全性で, マルチステージの安全性は複数の攻撃者に対して安全性を評価するゲームの安全性で, 考えている安全性 (ここでは, $\mathcal{G}$ と書くことにする) がシングルステージで定義されるならば, 任意の暗号アルゴリズム $\mathcal{C}$ に対して, $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$ 安全性は IFRO 安全な H^{P} を用いた $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ に対しても成り立つが, $\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}$ 安全性で $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ ではないマルチステージで定義される $\mathcal{G}$ と $\mathcal{C}$ が存在する.

そして, IFRO 安全性の代替案として, 任意の安全性をカバーする Reset Indifferentiability を用いた安全性が提案されている [51]. Reset Indifferentiability では, RIFRO (reset indifferentiable from $\mathcal{RO}$) の安全性を考えることで, 任意の $\mathcal{G}$ と任意の $\mathcal{C}$ に対して, $\mathcal{C}^{\mathcal{RO}}$ が $\mathcal{G}$ 安全ならば RIFRO 安全な H^{P} を用いた $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ も $\mathcal{G}$ 安全となることを保障する. しかし, 多くの重要なハッシュ関数に取り入れられているワンパスな構造に対して, ワンパスな構造を持つ任意の H^{P} は RIFRO 安全性を満たさないことが示されるなど, これまで RIFRO 安全な H^{P} の存在は知られていないのが現状である [51].

本章では, 暗号学的なハッシュ関数を説明した後, Indifferentiability, IFRO 安全性とその関連研究, そして, Reset Indifferentiability とその関連研究を説明して, 最後に, IFRO 安全性に関する研究課題を述べる.

2.2 暗号学的なハッシュ関数

2.2.1 定義

暗号学的なハッシュ関数は, 任意長の値を受け取り, 固定長の値を出力する関数である. ハッシュ関数を以下の記号で書く.

- $H : \{0,1\}^* \to \{0,1\}^n$

2.2.2 安全性

ハッシュ関数の定義は以下の 3 つである.

- 一方向性: ハッシュ関数の出力値 z が与えられたとき, $H(m) = z$ となることなる m を

見つけることが困難.

- 第二原像計算困難性: m が与えられたとき, $H(m) = H(m')$ となることなる m とは異なる値 m' を見つけることが困難.
- 衝突困難性: $H(m) = H(m')$ となることなる 2 つの値 m, m' を見つけることが困難.

ここで, “困難” とは H の総当たり攻撃の計算量より小さい計算量で上記の性質を破る攻撃が存在しないことである. また, 性質 X を破る攻撃が存在しないことは, X を破る攻撃者が存在しないことである. 以下, 計算量と攻撃者の 2 つの概念を入れて一方向性, 第二原像計算困難性, 衝突困難性を定義する.

定義 1 (一方向性) ハッシュ関数 H が一方向性を満たすとは, 任意の $z \in \{0,1\}^n$ が与えられた時, $z = H(m)$ となる m を 2^n より小さい計算量で見つける攻撃者が存在しないことである.

定義 2 (第二原像計算困難性) ハッシュ関数 H が第二原像計算困難性を満たすとは, 任意の $m \in \{0,1\}^*$ が与えられた時, $m \neq m'$ かつ $H(m) = H(m')$ となる m' を 2^n より小さい計算量で見つける攻撃者が存在しないことである.

定義 3 (衝突困難性) ハッシュ関数 H が衝突困難性を満たすとは, $m \neq m'$ かつ $H(m) = H(m')$ となる m, m' を $2^{n/2}$ より小さい計算量で見つける攻撃者が存在しないことである.

備考 1 誕生日解析により, 任意のハッシュ関数 H に対して, $2^{n/2}$ の計算量をかけると $m \neq m'$ かつ $H(m) = H(m')$ となる m, m' を見つけることができるため, 衝突困難性の総当たり攻撃の計算量は 2^n ではなく $2^{n/2}$ である. 詳しくは付録を参照されたい.

2.2.3 ハッシュ関数の設計

次に, H の具体的な構成法について説明する. ハッシュ関数 H は定義域拡張構造 H とプリミティブ P から構成される.

- プリミティブ P は入出力長が固定の関数である.
- 定義域拡張構造 H はプリミティブ P を用いて任意長の入力を固定長の値に圧縮する構造である.

H に P を組み込んだハッシュ関数を H^{P} と書くことにして, 以降の説明では, ハッシュ関数を H の代わりに H^{P} と書くことにする.

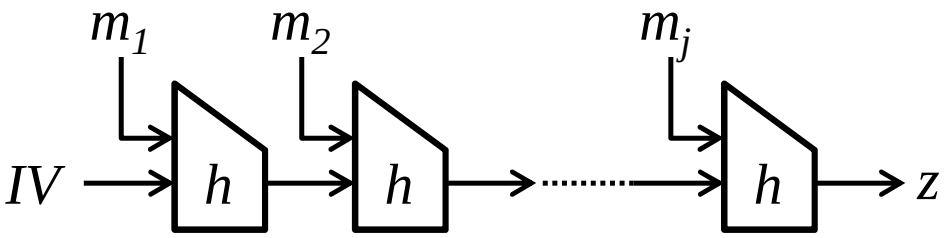

図.2.1　Merkle-Damgård 構造

ここで, 定義域拡張構造の例として, SHA-256 や SHA-512 [46] などで採用されている Merkle-Damgård (MD) 構造 [25, 41], SHA-512/224 や SHA-512/256 [46] で採用されている Chop Merkle-Damgård (ChopMD) 構造 [24], SHA-3 ハッシュ関数族 [45] に採用されている Sponge 構造を説明する [11].

MD 構造 [25, 41]

MD 構造はプリミティブとして入力長が $n+r$ bit, 出力長が n の圧縮関数 h を用いて任意長の入力 m を n bit の固定長 z に圧縮する構造である. $\mathsf{MD}^h(m)$ (または z) の計算方法は以下の通りである.

- 入力: $m \in \{0,1\}^*$
- 出力: $z \in \{0,1\}^n$
- 以下の手続きにより z を計算する. この手続きでは, 単射関数 $\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^r)^*$ と n bit の固定値 IV を用いる.
  1. $m^* \leftarrow \mathsf{pad}(m)$.
  2. $(m_1, \ldots, m_j) \leftarrow div(r, m^*)$.
  3. $s_0 \leftarrow IV$.
  4. for $i = 1, \ldots, j$ do $s_i \leftarrow h(s_{i-1}, m_i)$.
  5. $z \leftarrow s_j$.

図 2.1 は MD 構造の Step 3-5 の図である.

MD 構造は, pad が suffix-free を満たす関数で h が衝突困難性を満たすならば, MD^h も衝突困難性を満たすことが証明されている [25, 41]. よって, 小さい関数 h を衝突困難性を満たすように設計すれば, MD 構造を用いることで衝突困難性を満たすハッシュ関数が得られる. ここで, suffix-free な単射関数を suffix-free 関数と呼ぶことにする. suffix-free の定義は以下の

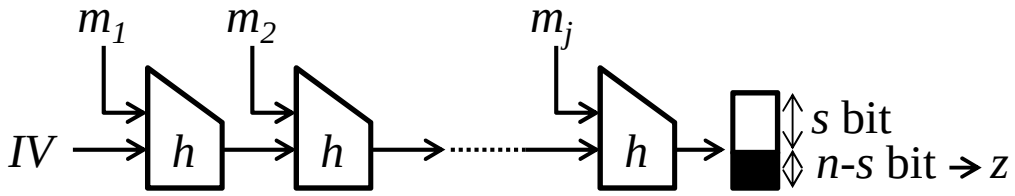

図.2.2　ChopMD 構造

通りである.

定義 4 pad : $\{0,1\}^* \to (\{0,1\}^*)^n$ が suffix-free とは, $m \neq m'$ となる任意の m, m' に対して, pad(m) が pad(m') の接尾辞とならないことである. すなわち, pad が suffix-free ならば pad(m') $= w \| \text{pad}(m)$ となる値 w が存在しない.

suffix-free な単射関数 pad を用いた MD 構造は strengthened MD 構造と呼ばれており, 以下, SMD 構造と呼ぶことにする.

suffix-free 関数の例として, m の後ろに 1 をパディングし, その後ろに 0 のビット列をパディングし, その後ろに m のビット長を 64 bit で表したビット列 $\langle m \rangle$ をパディングする pad(m) $:= m \| 1 \| 0^j \| \langle m \rangle$ とする関数がある. ここで, j は, 0 以上でかつ pad(m) の値が r の倍数となる最少の値である. また, $\langle m \rangle$ の例として, $m = 010$ の場合, $\langle m \rangle = 0 \cdots 011$ となる. 具体的な MD 構造の安全性の定理は以下の通りである.

定理 2.1 h が衝突困難性を満たすならば SMD^h も衝突困難性を満たす.

この証明は付録を参照されたい.

備考 2 MD 構造は理想的なハッシュ関数である $\mathcal{RO}$ との置き換えを考慮に入れて設計されていない. 一方で, $\mathcal{RO}$ との置き換えを考慮に入れて設計された構造として, ChopMD 構造と Sponge 構造がある. 以下, ChopMD 構造と Sponge 構造を説明する. $\mathcal{RO}$ との置き換えについては次章で説明する.

ChopMD 構造 [24]

ChopMD 構造は, n bit の入力 x のうち最上位の s bit を切り捨てて残りの $n - s$ bit を出力する $chop_s$ 関数を MD 構造の出力に追加した構造で, ChopMD$^h(m) := chop_s(\text{MD}^h(m))$ と定義される. 図 2.2 は ChopMD 構造の pad 関数を除いた図である.

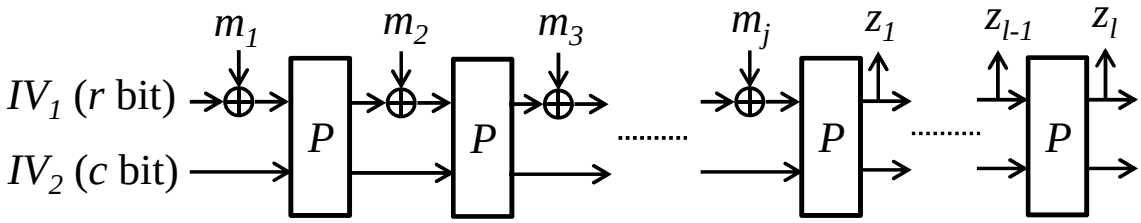

図.2.3　Sponge 構造

Sponge 構造 [11]

Sponge 構造はプリミティブとして $r+c$ bit の置換 P を用いた構造である [11]. Sponge 構造は出力長が可変長となっている構造で, 入力値としてメッセージ m と出力長 n を受け取り, n bit の値を出力する構造である. 具体的な仕様は以下の通りである.

- 入力: $m \in \{0,1\}^*$, 整数 n
- 出力: $z \in \{0,1\}^n$
- 以下の手続きにより出力値 z を計算する. この手続きでは, 出力値の最後の r bit が 0^r ではない単射関数 pad : $\{0,1\}^* \to (\{0,1\}^r)^*$ と r bit の固定値 IV_1 と c bit の固定値 IV_2 を用いる.
  1. $m^* \leftarrow \mathsf{pad}(m)$.
  2. $(m_1, \ldots, m_j) \leftarrow div(r, m^*)$.
  3. $s \leftarrow IV_1 \| IV_2$.
  4. for $i = 1, \ldots, j$ do $s \leftarrow P(s \oplus (m_i \| 0^c))$
  5. $z_1 \leftarrow s[1, r]$; $l \leftarrow \lceil n/r \rceil$
  6. for $i = 2, \ldots, l$ do $s \leftarrow P(s)$; $z_i \leftarrow s[1, r]$
  7. $z \leftarrow (z_1 \| \cdots \| z_{\lceil n/r \rceil})[1, n]$

図 2.3 は Sponge 構造の pad 関数を除いた図である.

2.3　ランダムオラクル ($\mathcal{RO}$) 証明法

$\mathcal{RO}$ 証明法は暗号アルゴリズム $\mathcal{C}$ にハッシュ関数 H^{P} を用いた暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性を理想的なハッシュ関数 $\mathcal{RO}$ を用いた $\mathcal{C}^{\mathcal{RO}}$ で証明するというものである. この証明法の考え方は，暗号アルゴリズム $\mathcal{C}$ に構造的な欠陥があるなら，暗号システムを構成する部品である

H^{P} を $\mathcal{RO}$ に置き換えたとしてもその欠陥は見つかるであろうから，$\mathcal{C}^{\mathcal{RO}}$ の安全性証明をつけよう，というものである．

ここでは $\mathcal{RO}$ と $\mathcal{C}^{\mathcal{RO}}$ の安全性を定義するが, まずは “安全性” を定義する．

定義 5 いかなる攻撃者 $\mathcal{A}$ も $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性を破ることができないとき $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ は安全であるという．

ここで, 暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の想定する使い方によって, $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性の定義が変わってくる．安全性の定義は安全性ゲーム $\mathcal{G}$ で定義されて, $\mathcal{G}$ は $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ を破ろうとする攻撃者 $\mathcal{A}$ の動作環境と攻撃成功の条件を規定する．そして, $\mathcal{G}$ を用いて $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性を次のように定義する．

定義 6 $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ に対する安全性ゲームとして $\mathcal{G}$ を考えて, $\mathcal{G}$ の環境で動作する任意の $\mathcal{A}$ が $\mathcal{G}$ で規定された攻撃成功の条件を無視できる確率を除いて満たすことができなければ, $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ は $\mathcal{G}$-安全であるという．

次に, $\mathcal{RO}$ を定義する．

定義 7 $\mathcal{L}$ を空集合とする．そして, $\mathcal{RO}$ へのクエリ m に対して, n bit の出力を以下のように定義する．

- If $\exists(m, z) \in \mathcal{L}$ then
  - z を出力する．
- Else
  - $z \xleftarrow{\$} \{0,1\}^n$.
  - (m, z) をハッシュリスト $\mathcal{L}$ に記録する．
  - z を出力する．

♦

最後に, $\mathcal{RO}$ 証明法を定義する．

定義 8 $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$ 安全性を $\mathcal{RO}$ 証明法で証明する場合, $\mathcal{C}$ に $\mathcal{RO}$ を用いた $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$ 安全性を考えて, 安全性ゲーム $\mathcal{G}$ では, 攻撃者 $\mathcal{A}$ や $\mathcal{C}$ など安全性ゲームに登場する全てのパーティーは $\mathcal{RO}$ にアクセスすることのみで関数値を知ることができるモデルを考える．このモデルをランダムオラクル ($\mathcal{RO}$) モデルと呼ぶ．そして, いかなる攻撃者 $\mathcal{A}$ も $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$ 安全性を破ることができないことを証明する．

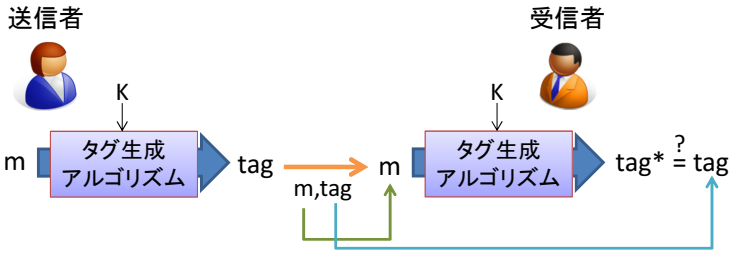

図.2.4　MAC アルゴリズムを用いたプロトコル

2.4　$\mathcal{RO}$ を用いた MAC の安全性

ここでは, $\mathcal{RO}$ 証明法の例として, メッセージ認証アルゴリズム (Message Authentication Code (MAC)) にハッシュ関数を用いた暗号システムの安全性を $\mathcal{RO}$ 証明法を用いて証明する. 具体的には, $\mathcal{C}$ として Secret Prefix MAC を, $\mathcal{G}$ として Unforgeability を考えて, Secret Prefix MAC が $\mathcal{RO}$ モデルで Unforgeability の安全性を満たすことを証明する.

まずは, MAC アルゴリズムの定義を説明する. 以下, MAC アルゴリズムを “アルゴリズム” を省略して MAC と呼ぶことにする. MAC は共通鍵暗号アルゴリズムの 1 つで, 以下の 2 つのアルゴリズムから構成される. ここで, 鍵サイズを k bit とする.

- 鍵生成アルゴリズム $KGen$: 鍵 K を $\{0,1\}^k$ からランダムに選んで, K を返す. $K \leftarrow KGen$ と書くことにする.
- タグ生成アルゴリズム $\mathcal{F}$: 鍵 K を用いてメッセージ m に対して, タグ tag を生成する. $tag \leftarrow \mathcal{F}_K(m)$ と書くことにする.

図 2.4 のように, MAC を用いると, メッセージ m に対して, タグ生成アルゴリズムを用いて tag を生成し, 秘密鍵 K を知っている人は (m, tag) のペアからタグ生成アルゴリズムを動かして m のタグ tag^* を生成し, $tag = tag^*$ となっていることを確認することで m の正しさを確認することができる.

MAC の安全性である Unforgeability のゲームを定義する. Unforgeability のゲームは, 秘密鍵 K を知らない人は正しいメッセージとタグのペアを生成できないことを規定したゲームである. 具体的な Unforgeability のゲームを以下に定義する.

- $K \leftarrow KGen$.
- 攻撃者 $\mathcal{A}$ は $\mathcal{F}_K$ にメッセージ m を質問し, それに対応するタグ tag を得ることができ

る. ここで, $\mathcal{F}_K$ を MAC オラクル, MAC オラクルへのクエリを MAC クエリと呼ぶことにする.

- $\mathcal{A}$ はメッセージとタグのペア (m^*, tag^*) を出力する
- $\mathcal{A}$ が m^* を $\mathcal{F}_K$ に問い合わせておらずかつ $tag^* = \mathcal{F}_K(m^*)$ ならば $\mathcal{A}$ の勝ちとする.

次に, Secret Prefix MAC を定義する. ハッシュ関数 H^{P} を用いた Secret Prefix MAC のタグ生成アルゴリズムは $\mathcal{F}_K(m) := \mathsf{H}^{\mathsf{P}}(K\|m)$ である.

最後に, ハッシュ関数 H^{P} を用いた Secret Prefix MAC の Unforgeability の安全性を $\mathcal{RO}$ 証明法を用いて証明する.

定理 2.2 任意の攻撃者が $\mathcal{RO}$ モデルで Secret Prefix MAC の Unforgeability を破る確率は, $\mathcal{A}$ の MAC クエリの回数の上限値を q とすると, $\max\{q/2^k, 1/2^n\}$ である.

証明. $\mathcal{A}$ を Unforgeability のゲームに沿って動作する任意の攻撃者とする. そして, m^*, tag^* を $\mathcal{A}$ が出力するメッセージとタグとする.

Unforgeability のゲームの条件から, MAC クエリの中に m^* は含まないため, $tag^* = \mathcal{RO}(K\|m^*)$ となるケースは以下の 2 つのケースのいずれかである.

- ケース 1: $\mathcal{A}$ が K を知っていて $\mathcal{RO}$ に $K\|m^*$ を問い合わせて tag^* を入手するケース.
- ケース 2: 偶然 $tag^* = \mathcal{RO}(K\|m^*)$ となる (m^*, tag^*) を出力するケース.

ケース 1 の場合, K は $\{0,1\}^k$ からランダムに選ばれて $\mathcal{A}$ に対して秘密となっているため, K を入手するためには, K を推定して MAC クエリで推定した K の正しさを確認しなければならない. q 回の MAC クエリで, q 回 K の正しさを確認することができるため, K を推定して当てることができる確率は $q/2^k$ 以下となる.

ケース 2 の場合, $\mathcal{RO}$ は出力がランダムに選ばれるため, 偶然 $tag^* = \mathcal{RO}(K\|m^*)$ となる確率は $1/2^n$ となる.

以上より, 任意の攻撃者が $\mathcal{RO}$ モデルで Secret Prefix MAC の Unforgeability を破る確率は $\max\{q/2^k, 1/2^n\}$ となる.

■

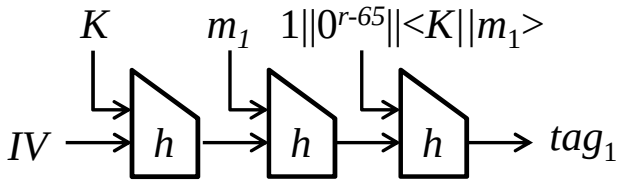

図.2.5　$\mathsf{SMD}^h(K\|m_1)$ の計算図

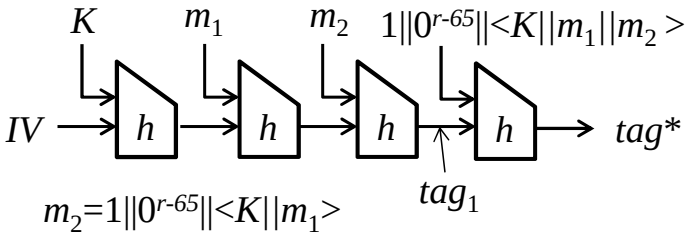

図.2.6　$\mathsf{SMD}^h(K\|m_1\|m_2)$ の計算図

2.5　MD 構造の否定的な結果

2.4 章では Secret Prefix MAC の Unforgeability の安全性を $\mathcal{RO}$ モデルで証明した．本章では, MD 構造に任意の圧縮関数 h を組み込んだハッシュ関数 MD^h を考えて, Secret Prefix MAC に MD^h を組み込んだ時に Unforgeability の安全性を満たさないことを示す．すなわち, $\mathcal{RO}$ 証明法で示された暗号アルゴリズムに MD 構造を用いたハッシュ関数を用いるは正しくないことが分かる．以下の証明では, MD 構造の 1 つのケースである SMD 構造について議論する．そして, suffix-free 関数 pad として $\mathsf{pad}(m) := m\|1\|0^j\|\langle m\rangle$ を用いた SMD 構造を扱う．

定理 2.3 (メッセージ長拡張攻撃) $r \geq 65$ とする．$|K| = r$, h を任意の圧縮関数として, $\mathcal{F}_K(m) = \mathsf{SMD}^h(K\|m)$ とする．$\mathcal{F}_K$ に対して, Unforgeability を確率 1 で破る攻撃者 $\mathcal{A}$ が存在する．ここで, SMD 構造の pad 関数を suffix-free 関数である $\mathsf{pad}(m) := m\|1\|0^j\|\langle m\rangle$ とする．

証明. $\mathcal{A}$ を以下に構成する．

- r bit のメッセージ m_1 を生成して, m_1 を MAC クエリして, タグ tag_1 を得る．ここで, $tag_1 = \mathsf{SMD}^h(K\|m_1)$ である．図 2.5 は $\mathsf{MD}^h(K\|m_1)$ の内部の計算図である．

- $m_2 := 1\|0^{r-65}\|\langle K\|m_1\rangle$ とする. $\langle K\|m_1\rangle$ は $K\|m_1$ のビット長を知っていれば求まることに注意されたい. また, m_2 は $\mathsf{MD}^h(K\|m_1)$ の計算で出てくる $\mathsf{pad}(K\|m_1)$ で扱うパディングされる値で, $\mathsf{pad}(K\|m_1) = K\|m_1\|m_2$ となっている.
- $tag^* \leftarrow h(tag, 1\|0^{r-65}\|\langle K\|m_1\|m_2\rangle)$ を計算する. ここで, $1\|0^{r-65}\|\langle K\|m_1\|m_2\rangle$ は $\mathsf{MD}^h(K\|m_1\|m_2)$ の計算で出てくる $\mathsf{pad}(K\|m_1\|m_2)$ で扱うパディングされる値で, $\mathsf{pad}(K\|m_1) = K\|m_1\|m_2\|1\|0^{r-65}\|\langle K\|m_1\|m_2\rangle$ となっている.
- $m^* := m_1\|m_2$ として, メッセージとタグのペア (m^*, tag^*) を出力する. 図 2.6 は $\mathsf{SMD}^h(K\|m^*)$ の内部の計算図である.

以上より, 確率 1 で $tag^* = \mathsf{SMD}^h(K\|m^*)$ となるため上記の定理が証明できた.

■

上記の攻撃はメッセージ長拡張攻撃と呼ばれる攻撃法で, MD 構造の h を繰り返す構造を利用して, $\mathsf{SMD}^h(m)$ の z を知っていれば m を知らずに $\mathsf{SMD}^h(m\|m')$ の出力値を計算できることを利用した攻撃法である. m から $m\|m'$ にメッセージ長を拡張した攻撃法なので, メッセージ長拡張攻撃と呼ばれている.

2.6 IFRO 安全性

2.5 章では, $\mathcal{RO}$ を用いて安全性証明がされた暗号アルゴリズム $\mathcal{C}$ のハッシュ関数に MD 構造を持つハッシュ関数を用いることは正しくないことを示した. 本章では, $\mathcal{RO}$ をハッシュ関数 H^{P} で置き換えることを保証するハッシュ関数の (必要十分) 条件である IFRO 安全性を説明する.

2.6.1 Indifferentiability の理論

IFRO 安全性は Maurer らによって提案された Indifferentiability 理論を用いた安全性である [40]. まず, Indifferentiability を説明する.

Indifferentiability を用いると, 安全性ゲーム $\mathcal{G}$, 暗号アルゴリズム $\mathcal{C}$, 2 つの暗号部品 $\mathcal{U}, \mathcal{V}$ に対して, $\mathcal{C}^{\mathcal{U}}$ の $\mathcal{G}$-安全性から $\mathcal{C}^{\mathcal{V}}$ の $\mathcal{G}$-安全性を保障することができる.

Indifferentiability では, 攻撃者がアクセスする暗号部品のインターフェースとそれ以外のパーティーがアクセスする暗号部品のインターフェースを分けて考える. まず, 暗号部品のインターフェースの記号として, $\mathcal{U}$ が攻撃者に対して公開しているインターフェースを $\mathcal{U}.adv$ と

書くことにする．このインターフェースを adversarial インターフェースと呼ぶ．そして, $\mathcal{U}$ がそれ以外のパーティー, 例えば暗号アルゴリズム, に対して公開しているインターフェースを $\mathcal{U}.hon$ と書くことにする．このインターフェースを honest インターフェースと呼ぶ．暗号システム $\mathcal{C}^{\mathcal{U}}$ に対する安全性ゲーム $\mathcal{G}$ で, 攻撃者は $\mathcal{U}.adv$ にアクセスして, それ以外のパーティー, 例えば暗号アルゴリズム $\mathcal{C}$ や $\mathcal{G}$ のチャレンジャー, は $\mathcal{U}.hon$ にアクセスする．以下, 暗号システム $\mathcal{C}^{\mathcal{U}}$ を考える場合, インターフェースに注目した議論する場合は $\mathcal{C}^{\mathcal{U}.hon}$ と hon を明示する．また, $\mathcal{V}$ のインターフェースも同様に定義する．

ここで, Indifferentiability では, 暗号システム $\mathcal{C}^{\mathcal{V}}$ の $\mathcal{G}$-安全性から暗号システム $\mathcal{C}^{\mathcal{U}}$ の $\mathcal{G}$-安全性を保障することを, $\mathcal{C}^{\mathcal{U}}$ に対する安全性ゲーム $\mathcal{G}$ の任意の攻撃者 $\mathcal{A}$ に対して, $\mathcal{C}^{\mathcal{V}}$ に対する安全性ゲーム $\mathcal{G}$ の攻撃者 $\mathcal{B}$ が構成できることと定義する．そして, このことが示せると, $\mathcal{C}^{\mathcal{V}}$ に対する $\mathcal{G}$ の攻撃者 $\mathcal{B}$ が存在しないならば, $\mathcal{C}^{\mathcal{U}}$ に対する $\mathcal{G}$ の攻撃者 $\mathcal{A}$ も存在しないことが保証できるため, $\mathcal{C}^{\mathcal{U}}$ が $\mathcal{G}$-安全性ならば $\mathcal{C}^{\mathcal{U}}$ も $\mathcal{G}$-安全性となる．

具体的には, $\mathcal{C}^{\mathcal{V}}$ が $\mathcal{G}$-安全性ならば $\mathcal{C}^{\mathcal{U}}$ も $\mathcal{G}$-安全性であることを以下のように定義する．

定義 9 安全性ゲーム $\mathcal{G}$ を 1 bit を出すゲームとする．そして, $\mathcal{C}^{\mathcal{U}}$ に対する $\mathcal{G}$ の環境に沿って動作する攻撃者 $\mathcal{A}$ がゲームに勝つイベントを $\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1$ と書くことにする．

そして, 安全性ゲーム $\mathcal{G}$ で動作する $\mathcal{C}^{\mathcal{U}}$ に対する任意の攻撃者 $\mathcal{A}$ に対して, 以下の確率の差が無視できるくらい小さい値で抑えられる $\mathcal{C}^{\mathcal{V}}$ に対する攻撃者 $\mathcal{B}$ が存在するならば, $\mathcal{C}^{\mathcal{U}}$ の $\mathcal{G}$-安全性は $\mathcal{C}^{\mathcal{V}}$ の $\mathcal{G}$-安全性と同じ安全性を担保すると定義して, $\mathcal{C}^{\mathcal{U}} \succ \mathcal{C}^{\mathcal{V}}$ と書くことにする．

$$\Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1]$$

♦

ここで, Indifferentiability では, $\mathcal{U}, \mathcal{V}$ に対して, 以下の安全性を考える．

定義 10 1 bit の値を出力する任意の識別者 $\mathcal{D}$ に対して, 以下のアドバンテージが無視できるくらい小さい確率で抑えられるシミュレータ S が存在するならば, "$\mathcal{U}$ is indifferentiable from $\mathcal{V}$" と定義して, これを $\mathcal{U} \sqsubset \mathcal{V}$ と書くことにする．

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathcal{U},\mathcal{V},S}(\mathcal{D}) := \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1]$$

♦

上記の定義は, $\mathcal{D}$ が 2 つのオラクル (L, R) にアクセスして, $(L, R) = (\mathcal{U}.hon, \mathcal{U}.adv)$ のケースと $(L, R) = (\mathcal{V}.hon, S^{\mathcal{V}.adv})$ のケースを識別するゲームととらえることもでき

る．以降では, L を L オラクル (ライトオラクル) と呼び, R を R オラクル (レフトオラクル) と呼ぶことにする．そして, $(L,R) = (\mathcal{U}.hon,\mathcal{U}.adv)$ のケースを Real World と呼び, $(L,R) = (\mathcal{V}.hon, S^{\mathcal{V}.adv})$ のケースを Ideal World と呼ぶことにする．

ここで, $\mathcal{U} \sqsubset \mathcal{V}$ は, $(\mathcal{U}.hon,\mathcal{U}.adv)$ から得られる任意の情報は, シミュレータ S を介して $(\mathcal{V}.hon,\mathcal{V}.adv)$ から得ることができることを保証する．すなわち, 安全性ゲーム $\mathcal{G}$ と暗号アルゴリズム $\mathcal{C}$ を考えて, $\mathcal{U}$ を用いた暗号システム $\mathcal{C}^{\mathcal{U}}$ に対する攻撃者を $\mathcal{A}$, $\mathcal{V}$ を用いた暗号システム $\mathcal{C}^{\mathcal{V}}$ に対する攻撃者を $\mathcal{B}$ とすると, $\mathcal{U} \sqsubset \mathcal{V}$ から, $\mathcal{A}$ が $(\mathcal{C}^{\mathcal{U}.hon},\mathcal{U}.adv)$ から得られる任意の情報は, $\mathcal{B}$ が $\mathcal{A}$ と S の手続きを利用すると $(\mathcal{C}^{\mathcal{V}.hon},\mathcal{V}.adv)$ から得ることができる．すなわち, 攻撃者が $(\mathcal{C}^{\mathcal{U}.hon},\mathcal{U}.adv)$ から得られる情報は, 攻撃者が $(\mathcal{C}^{\mathcal{V}.hon},\mathcal{V}.adv)$ から得られる情報以上の情報はないことが保証できる．よって, $\mathcal{U} \sqsubset \mathcal{V}$ ならば, 任意の $\mathcal{B}$ に対して $\mathcal{C}^{\mathcal{U}}$ が $\mathcal{G}$-安全となる場合, 任意の $\mathcal{A}$ に対して $\mathcal{C}^{\mathcal{V}}$ が $\mathcal{G}$-安全となることが保証できる．具体的には, 以下の定理が成り立つ [40, 51]．ここで, 任意の暗号アルゴリズムの集合を $\mathbf{C}$, 任意のシングルステージゲームの集合を $\mathbf{G}_s$ と書くことにする．シングルステージゲームとは攻撃者の人数が 1 人を想定したゲームのことである．

定理 2.4 $\mathcal{U} \sqsubset \mathcal{V} \Leftrightarrow \forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathcal{U}} \succ \mathcal{C}^{\mathcal{V}}$.

具体的には,

$$\forall \mathcal{D}, \exists S : \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$
$$\Leftrightarrow$$
$$\forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

証明. まず $(\Rightarrow)$ を証明する．$\mathcal{U} \sqsubset \mathcal{V}$ を仮定する．すなわち, 無視できるくらい小さい値 ϵ に対して, 以下の式が成り立つ．

$$\forall \mathcal{D}, \exists S : \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

ここで, $\forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}$ に対して, $\mathcal{D}^{L,R} := \mathcal{G}^{\mathcal{C}^L,\mathcal{A}^R}$ とすると, 以下の式が成り立つ．

$$\Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{A}^{S^{\mathcal{V}.adv}}} \Rightarrow 1] < \epsilon$$

そして, $\mathcal{B}^{\mathcal{V}.adv} := \mathcal{A}^{S^{\mathcal{V}.adv}}$ とすると, 以下の式が成り立つ．

$$\forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

以上より, $(\Rightarrow)$ が示せた．

次に ($\Leftarrow$) を示す. $\forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathcal{U}} \succ \mathcal{C}^{\mathcal{V}}$ を仮定する. すなわち, 無視できるくらい小さい値 ϵ に対して, 以下の式が成り立つ.

$$\forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon}, \mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon}, \mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

ここで, $\mathcal{G} := \mathcal{D}$, $\mathcal{C}^L := L$, $\mathcal{A}^{\mathcal{U}.adv} := \mathcal{U}.adv$, $\mathcal{B} := S$ とすると, 以下の式が成り立つ.

$$\forall \mathcal{D}, \exists S : \Pr[\mathcal{D}^{\mathcal{U}.hon, \mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon, S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

以上より, ($\Leftarrow$) が示せた.

■

備考 3 上記の定理は Indifferentiability が任意のシングルステージゲームで $\mathcal{V}$ から $\mathcal{U}$ への置き換えを保証するものの, マルチステージゲームで $\mathcal{V}$ から $\mathcal{U}$ への置き換えを保証できないことが知られている [51]. この内容は, 2.7 章で議論する.

2.6.2 IFRO 安全性の定義

ハッシュ関数の設計に，Indifferentiability を適用すると，$\mathcal{RO}$ からプリミティブ P を理想化したハッシュ関数 H^{P} への置き換えを保証する IFRO (Indifferentiability from $\mathcal{RO}$) 安全性を定義することができる. IFRO 安全性は, P を理想化して考えた定義域拡張構造 H に注目した安全性なので, IFRO 安全性は H に対する安全性である.

ここで, 暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の安全性ゲームは honest なパーティーである暗号アルゴリズム $\mathcal{C}$ が H^{P} にアクセスして, 攻撃者が P にアクセスするゲームである. よって, Real World では $(L, R) = (\mathsf{H}^{\mathsf{P}}, \mathsf{P})$ となる. また, $\mathcal{RO}$ は全てのパーティーがアクセス可能なオラクルなので, $\mathcal{RO}$ の honest と adversarial インターフェースはともに $\mathcal{RO}$ そのものとなる. すなわち, Ideal World では $(L, R) = (\mathcal{RO}, S^{\mathcal{RO}})$ となる. H^{P} と $\mathcal{RO}$ の Indifferentiability を考えた定義は以下の通りである.

定義 11 (IFRO 安全性) 1 bit の値を出力する任意の識別者 $\mathcal{D}$ に対して, 以下のアドバンテージが無視できるくらい小さい確率で抑えられるシミュレータ S が存在するならば, H^{P} は IFRO 安全で (Indifferentiable from $\mathcal{RO}$), $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO}$ と書くことにする.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{H}^{\mathsf{P}}, \mathcal{RO}, S}(\mathcal{D}) := \Pr[\mathcal{D}^{\mathsf{H}^{\mathsf{P}}, \mathsf{P}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{RO}, S^{\mathcal{RO}}} \Rightarrow 1]$$

♦

そして定理 2.4 によって，以下の系が保証できる．

系 1 $\mathsf{H}^{\mathsf{P}} \sqsubset \mathcal{RO} \Leftrightarrow \forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ \mathcal{C}^{\mathcal{RO}}$. ♦

2.6.2.1 IFRO 安全な定義域拡張構造

ここで, IFRO 安全性が提案される前に提案された MD 構造は, 定理 2.3 と系 1 を組み合わせることで, 任意の h に対して, $\mathsf{MD}^h \not\sqsubset \mathcal{RO}$ となることを示すことができる. 具体的な識別攻撃法の手順は定理 2.3 で示した, メッセージ長拡張攻撃により $\mathsf{MD}^h \not\sqsubset \mathcal{RO}$ となることを示すことができる.

一方, ChopMD 構造や Sponge 構造は内部変数の値を一部切り落として, 残りの bit を出力する構造を持つため, これらの構造にメッセージ長拡張攻撃を適用するためには, 切り落とした bit を当てる必要がある. そして, この切り落とした bit は乱数値となっているため, この bit を当てることは困難となり, ChopMD 構造や Sponge 構造に対してメッセージ長拡張攻撃を適用できないことが示せて, IFRO 安全性を満たすことを証明することができる.

付録に, IFRO 安全性の証明方法を具体的に説明し, 例として, 簡易版の ChopMD 構造の IFRO 安全性の証明を載せておく.

2.7 Indifferentiability の否定的な結果

2.6 章で IFRO 安全性となるハッシュ関数 H^{P} を設計すると, 任意のシングルステージゲームで $\mathcal{RO}$ から H^{P} への置き換えができることを説明した. 本章では, 文献 [51] で示された, マルチステージゲームで $\mathcal{RO}$ から IFRO 安全な H^{P} への置き換えができないことを説明する. これは, マルチステージゲームで定義された安全性について, $\mathcal{RO}$ 証明法で安全性が証明された暗号アルゴリズムに IFRO 安全な H^{P} を組み込むのは正しくないことを意味する.

マルチステージゲームの集合を $\mathbf{G}_m$ と書くことにする. マルチステージゲームとは, ゲームで考える攻撃者 $\mathcal{A}$ が複数の攻撃者 $(\mathcal{A}_1, \ldots, \mathcal{A}_i)$ から構成され, 攻撃者間でシェアできるステイトが制限されているゲームのことである. そして, 以下のことが成り立つ.

定理 2.5 $\mathbf{H}$ を IFRO 安全なハッシュ関数の集合とする. 次のことが成り立つ. $\exists \mathsf{H}^{\mathsf{P}} \in \mathbf{H}, \exists \mathcal{G} \in \mathbf{G}_m, \exists \mathcal{C} \in \mathbf{C}$ s.t. $\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}$ 安全だが $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ は $\mathcal{G}$ 安全ではない.

具体例として, 以下のケースを考える.

- $\mathsf{H}^{\mathsf{P}} = \mathsf{ChopMD}^h$

- $\mathcal{C}$ = ユーザとサーバのチャレンジレスポンスプロトコル $\mathcal{CR}$ [51]
- $\mathcal{G}$ = マルチステージゲームで定義される CRP ゲーム [51]

まず, $\mathcal{CR}$ にハッシュ関数 $\mathsf{H}^{\mathsf{P}} : \{0,1\}^* \to \{0,1\}^n$ を用いた暗号システム $\mathcal{CR}^{\mathsf{H}^{\mathsf{P}}}$ を定義する.

定義 12 (チャレンジレスポンスプロトコル $\mathcal{CR}^{\mathsf{H}^{\mathsf{P}}}$) A をユーザ, B をサーバとする. A と B のチャレンジレスポンスプロトコルを以下に定義する.

1. A は B に $4n$ bit のメッセージ m を送り, B は m を保存する.
2. A は B が m を記録していることを確認するために, $4n$ bit の乱数値 cha を選んで B に送る.
3. B は $\mathsf{H}^{\mathsf{P}}(m\|cha)$ の出力値 z を A に送る.
4. A は $z = \mathsf{H}^{\mathsf{P}}(m\|cha)$ となっていることを確認して, 等しければ B は m を記録していると判断する.

♦

次に, $\mathcal{CR}^{\mathsf{H}^{\mathsf{P}}}$ の安全性ゲームである CRP ゲームを定義する.

定義 13 (CRP ゲーム) ここで, 攻撃者を $\mathcal{B}_1, \mathcal{B}_2$ として, $\mathcal{B}_1$ から $\mathcal{B}_2$ に引き継げるステイト (または値) のサイズは $2n$ bit 以下とする.

1. チャレンジャーは $\mathcal{B}_1$ に $4n$ bit の乱数値 m を送る.
2. $\mathcal{B}_1$ は $2n$ bit 以下の値 st を出力する.
3. $\mathcal{B}_2$ は st を受け取る.
4. チャレンジャーは $4n$ bit の乱数値 cha を選んで $\mathcal{B}_2$ に送る.
5. z をチャレンジャーに送る.
6. チャレンジャーは z と $\mathsf{H}^{\mathsf{P}}(m\|cha)$ を比較して, 等しければ攻撃者の勝ちとする.

♦

次に, $\mathcal{CR}$ が $\mathcal{RO}$ モデルで CRP-安全で, IFRO 安全なハッシュ関数を用いた時に CRP-安全でないことを示す.

補題 1 $\mathcal{CR}^{\mathcal{RO}}$ は CRP-安全である. ♦

証明. $\mathcal{CR}^{\mathcal{RO}}$ に対して CRP ゲームの攻撃者 $(\mathcal{B}_1, \mathcal{B}_2)$ の戦略は以下の 2 つに分けることができる.

- $\mathcal{B}_2$ が m を復元して $z = \mathcal{RO}(m\|cha)$ となる z を出力するケース: st の長さは $2n$ bit なので, $\mathcal{B}_1$ から $\mathcal{B}_2$ に渡せる m の bit 長は $2n$ bit 以下である. そして, $\mathcal{B}_2$ は残りの $2n$ bit を当てる必要があるため, m は乱数なので, $\mathcal{B}_2$ が残りの $2n$ bit を当てる確率は $1/2^{2n}$ である.
- それ以外のケース: このケースは $\mathcal{B}_2$ が $\mathcal{RO}(m\|cha)$ の n bit の出力値を推定するケースで, 推定が当たる確率は $1/2^n$ である.

以上より, 攻撃者がいかなる戦略をとても, 攻撃者が勝つ確率は無視できるくらい小さいので $\mathcal{CR}^{\mathcal{RO}}$ は CRP 安全である.

■

補題 2 $h : \{0,1\}^{6n} \to \{0,1\}^{2n}$ をランダムオラクル圧縮関数として, ChopMD 構造に h を組み込んだ IFRO 安全なハッシュ関数 ChopMD^h を考える. ここで, $chop_s$ は $2n$ bit のうち半分の n bit を切り落として, 残りの n bit を出力する構造を考える. この場合, $\mathcal{CR}^{\mathsf{ChopMD}^h}$ は CRP-安全はない. ♦

証明. ここで説明を簡単にするために, pad 関数を取り除いた ChopMD^h を説明する. すなわち, $\mathsf{ChopMD}^h(m\|cha) = chop_n(h(h(IV, m), cha)$ となる. そして, このケースで CRP ゲームに確率 1 で勝つことができる攻撃者が存在することを示す.

- $\mathcal{B}_1$ の手続き
  1. $4n$ bit の値 m を受け取る.
  2. $st \leftarrow h(IV, m)$ を計算する.
  3. st を出力する.
- $\mathcal{B}_2$ の手続き
  1. st と $4n$ bit の値 cha を受け取る.
  2. $z \leftarrow chop_n(h(st, cha))$ を計算する.
  3. z を出力する.

以上の手続きを事項することにより, 確率 1 で $z = \mathsf{ChopMD}^h(m\|cha)$ となり, $\mathcal{CR}^{\mathsf{ChopMD}^h}$ は CRP-安全はないとはならないことが示せた.

■

上記の定理は ChopMD^h の中間値 $h(IV, m)$ を知っていれば m を知らずに $\mathsf{ChopMD}(m\|m')$

を計算することができる性質を利用した攻撃である．この性質は ChopMD^h 以外にも Sponge もこのような性質を持つ．そして，この性質を持つハッシュ関数はワンパスハッシュ関数と呼ばれている．以下にワンパスハッシュ関数を具体的に定義する．

定義 14 (ワンパスハッシュ関数) ある全単射関数 f と任意の m に対して，$|m| = |m_1| + |m_2|$，$(m_1, m_2) = f(m)$ で $\mathsf{H}^{\mathsf{P}}(m) = \mathsf{H}_2^{\mathsf{P}}(\mathsf{H}_1^{\mathsf{P}}(m_1) \| m_2)$ と書ける $\mathsf{H}_1^{\mathsf{P}}, \mathsf{H}_2^{\mathsf{P}}$ が存在する H^{P} をワンパスハッシュ関数と呼ぶ．

そして，ChopMD^h と同様の攻撃がワンパスハッシュ関数に適用できるため，以下の補題が成り立つ．

補題 3 H^{P} を任意のワンパスハッシュ関数とする．この場合，$\mathcal{CR}^{\mathsf{H}^{\mathsf{P}}}$ は CRP 安全ではない．

2.8 Reset Indifferentiability

2.7 章では，$\mathcal{RO}$ を用いてシングルステージゲームの安全性について証明がされた暗号アルゴリズム $\mathcal{C}$ のハッシュ関数に IFRO 安全性を満たすハッシュ関数を用いることは正しくないことを示した．本章では，マルチステージゲームを含む任意の安全性ゲームでの置き換えを保証した Reset Indifferentiability を説明する [51].

2.8.1 Reset Indifferentiability 理論

Reset Indifferentiability は，安全性ゲーム $\mathcal{G}$，暗号アルゴリズム $\mathcal{C}$，2 つの部品システム $\mathcal{U}, \mathcal{V}$ に対して，$\mathcal{C}^{\mathcal{U}}$ の $\mathcal{G}$-安全性から $\mathcal{C}^{\mathcal{V}}$ の $\mathcal{G}$-安全性を保証する理論である [51].

Indifferentiability では，定理 2.4 で示したように，任意の $\mathcal{D}$ が Real World $(L, R) = (\mathcal{U}.hon, \mathcal{U}.adv)$ と Ideal World $(L, R) = (\mathcal{V}.hon, S^{\mathcal{V}.adv})$ を識別できないステイトフルなシミュレータ S が存在すると，任意のシングルステージゲーム $\mathcal{G}_s$ と任意の暗号アルゴリズム $\mathcal{C}$，$\mathcal{G}_s$ ゲームの任意の攻撃者 $\mathcal{A}$ が得られる $(\mathcal{U}.hon, \mathcal{U}.adv)$ に関する情報は，“$\mathcal{B}^R := \mathcal{A}^{S^R}$” とすることで $(\mathcal{V}.hon, \mathcal{V}.adv)$ から同じ情報を得ることができる $\mathcal{B}$ を構成できて，結果，$\mathcal{C}^{\mathcal{U}}$ が $\mathcal{G}_s$-安全性ならば $\mathcal{C}^{\mathcal{V}}$ も $\mathcal{G}$-安全性となる．

一方で，この議論で出てくる “$\mathcal{B}^R := \mathcal{A}^{S^R}$” をマルチステージゲーム $\mathcal{G}_m$ に適用すると，$\mathcal{C}^{\mathcal{U}}$ に対する攻撃者を $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_i)$ とした場合，$(\mathcal{A}_1^{S^R}, \ldots, \mathcal{A}_i^{S^R})$ となる．ここで，マルチステージゲームは攻撃者間でシェアできるステイトの条件が制限されていることに注意すると，

$(\mathcal{A}_1^{S^R}, \ldots, \mathcal{A}_i^{S^R})$ は S のステイトをシェアする攻撃者となり, $\mathcal{C}^{\mathcal{V}}$ に対する攻撃者として定義することができない.

そこで, Reset Indifferentiability では, ステイトレスなシミュレータを考えることで, $(\mathcal{A}_1^{S^R}, \ldots, \mathcal{A}_i^{S^R})$ の S によるステイトの問題を解決した. S がステイトレスなシミュレータなので, $(\mathcal{A}_1^{S^R}, \ldots, \mathcal{A}_i^{S^R})$ を $\mathcal{C}^{\mathcal{V}}$ に対する攻撃者 $\mathcal{B}$ と定義できるため, 定理 2.4 の証明がマルチステージゲームでもワークするようになる. ここで, Reset Indifferentiability では, 具体的に, $\mathcal{U}, \mathcal{V}$ に対して, 以下の安全性を考える.

定義 15 1 bit の値を出力する任意の識別者 $\mathcal{D}$ に対して, 以下のアドバンテージが無視できるくらい小さい確率で抑えられる**ステイトレス**なシミュレータ S が存在するならば, "$\mathcal{U}$ is reset indifferentiable from $\mathcal{V}$" と定義して, これを $\mathcal{U} \sqsubset_r \mathcal{V}$ と書くことにする.

$$\mathsf{Adv}^{\mathsf{rindiff}}_{\mathcal{U},\mathcal{V},S}(\mathcal{D}) := \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1]$$

♦

そして, 以下の定理が成り立つ. ここで, $\mathbf{G}_m$ をマルチステージゲームの集合とする.

定理 2.6 $\mathcal{U} \sqsubset_r \mathcal{V} \Leftrightarrow \forall \mathcal{G} \in \mathbf{G}_s \cup \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathcal{U}} \succ \mathcal{C}^{\mathcal{V}}$.

具体的には,

$$\forall \mathcal{D}, \exists \text{ (stateless) } S : \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$
$$\Leftrightarrow$$
$$\forall \mathcal{G} \in \mathbf{G}_s \cup \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

証明. まず $(\Rightarrow)$ を証明する. $\mathcal{U} \sqsubset \mathcal{V}$ を仮定する. すなわち, 無視できるくらい小さい値 ϵ に対して, 以下の式が成り立つ.

$$\forall \mathcal{D}, \exists S : \Pr[\mathcal{D}^{\mathcal{U}.hon,\mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon,S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

そして, $\forall \mathcal{G} \in \mathbf{G}_s \cup \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}$ に対して, $\mathcal{D} := \mathcal{G}^{\mathcal{C}^L,\mathcal{A}^R}$ とすると, 以下の式が成り立つ. ここで, $\mathcal{A}$ の j ステージ目の攻撃者を $\mathcal{A}_j$ として, $\mathcal{B}$ の j ステージ目の攻撃者を $\mathcal{B}_j$ として, $\mathcal{A}^R = (\mathcal{A}_1^R, \ldots, \mathcal{A}_i^R)$, $\mathcal{B}^R = (\mathcal{B}_1^R, \ldots, \mathcal{B}_i^R)$ とする.

$$\Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon},\mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon},\mathcal{A}^{S^{\mathcal{V}.adv}}} \Rightarrow 1] < \epsilon$$

そして, S はステイトレスなので, マルチステージゲームのステイトの条件によらず各ステージの攻撃者 $\mathcal{B}_j$ の手続きとして組み込むことができるため, $\mathcal{B}^{\mathcal{V}.adv} := (\mathcal{A}_1^{S^{\mathcal{V}.adv}}, \ldots, \mathcal{A}_i^{S^{\mathcal{V}.adv}})$ とすることができて, そして以下の式が成り立つ.

$$\forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon}, \mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon}, \mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

以上より, ($\Rightarrow$) が示せた.

次に ($\Leftarrow$) を示す. $\forall \mathcal{G} \in \mathbf{G}_s \cup \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathcal{U}} \succ \mathcal{C}^{\mathcal{V}}$ を仮定する. すなわち, 無視できるくらい小さい値 ϵ に対して, 以下の式が成り立つと仮定する.

$$\forall \mathcal{G} \in \mathbf{G}_s \cup \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{A}, \exists \mathcal{B} \text{ s.t. } \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{U}.hon}, \mathcal{A}^{\mathcal{U}.adv}} \Rightarrow 1] - \Pr[\mathcal{G}^{\mathcal{C}^{\mathcal{V}.hon}, \mathcal{B}^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

ここで, $\mathcal{G} := \mathcal{D}$, $\mathcal{C}^L := L$, $\mathcal{A}^{\mathcal{U}.adv} := \mathcal{U}.adv$, $\mathcal{B} := S$ とすると, 以下の式が成り立つ.

$$\forall \mathcal{D}, \exists S : \Pr[\mathcal{D}^{\mathcal{U}.hon, \mathcal{U}.adv} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{V}.hon, S^{\mathcal{V}.adv}} \Rightarrow 1] < \epsilon$$

以上より, ($\Leftarrow$) が示せた.

■

備考 4 定理 2.6 から $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{RO}$ (Reset Indifferentiable from $\mathcal{RO}$ (RIFRO) 安全性) となるハッシュ関数 H^{P} が構成できれば, 任意のマルチステージゲームで $\mathcal{RO}$ から H^{P} への置き換えを保証できる. 一方で, 補題 3 と定理 2.6 から, 任意のワンパスハッシュ関数 H^{P} は RIFRO 安全性を満たさない. また, ワンパスハッシュ関数以外のハッシュ関数に対して, 補題 3 相当の結果は知られていないものの, 今のところ $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{RO}$ となる H^{P} は見つかっていない.

2.9 IFRO 安全性に関する研究課題

本章では IFRO 安全性に関する解決すべき問題について述べる.

2.9.1 研究課題 1

2.5 章で, $\mathcal{G}$ としてシングルステージで定義される安全性ゲームである Unforgeability を考えて, $\mathcal{C}$ として Secret Prefix MAC を考えた場合, MD 構造を用いたハッシュ関数 MD^h に対して $\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}$ 安全であるが $\mathcal{C}^{\mathsf{MD}^h}$ は $\mathcal{G}$ 安全ではないことを示した.

一方, この例は全てのシングルステージゲームである $\mathcal{G}_s$ に対して成り立つとは限らない. また, MD 構造は重要なハッシュ関数である SHA-256 や SHA-512 に採用されているため, 以下の課題は重要な研究課題である.

- $\mathcal{C}^{\mathcal{RO}}$ が $\mathcal{G}$ 安全のときに $\mathcal{C}^{\mathsf{MD}^h}$ が $\mathcal{G}$ 安全となる $\mathcal{G} \in \mathbf{G}_s, \mathcal{C} \in \mathbf{C}$ を見つける. 特に, $\mathcal{G}$ として重要な安全性ゲームである, 公開鍵暗号の安全性 IND-CCA やデジタル署名の安全性 EUF-CMA を扱い, そして, $\mathcal{C}$ として, 実用的な公開鍵暗号アルゴリズムである OAEP 暗号や RSA-KEM, 実用的なデジタル署名である FDH 署名に対して, 上記のことを示すことは重要である.

本研究では上記の研究課題を 3 章で議論する.

2.9.2 研究課題 2

2.7 章で, 安全性ゲームとして CRP, 暗号アルゴリズムとして $\mathcal{CR}$ を考えた場合, 任意の IFRO 安全なワンパスハッシュ関数 H^{P} に対して $\mathcal{C}^{\mathcal{RO}}$ は $\mathcal{G}$ 安全であるが $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ は $\mathcal{G}$ 安全ではないことを示した.

一方, この例は全てのマルチステージゲームで成り立つとは限らず, CRP 以外の安全性ゲーム $\mathcal{G}$ と $\mathcal{C}^{\mathcal{RO}}$ が $\mathcal{G}$ 安全な $\mathcal{CR}$ 以外の暗号アルゴリズム $\mathcal{C}$ に対して, $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ も同様に $\mathcal{G}$ 安全となる可能性がある. H^{P} として, Sponge 構造と ChopMD 構造は SHA-3 ハッシュ関数族と SHA-512/224 と SHA-512/256 にそれぞれ採用されているため, これらの構造をもつハッシュ関数を考えることは特に重要であるため, 以下の課題は重要な研究課題である.

- $\mathcal{C}^{\mathcal{RO}}$ が $\mathcal{G}$ 安全のときに $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ が $\mathcal{G}$ 安全となる $\mathcal{G} \in \mathbf{G}_m, \mathcal{C} \in \mathbf{C}$ を見つける. 特に, $\mathcal{G}$ として重要な安全性ゲームである, 検索可能暗号やヘッジ暗号の安全性ゲームである CDA ゲームを考えて, そして, $\mathcal{C}$ として, 実用的な公開鍵暗号アルゴリズムである REwH に対して, 上記のことを示すことは重要である.

本研究では上記の研究課題を 4 章で議論する.

2.9.3 研究課題 3

ハッシュ関数に求められる実装に関する要件として，ハッシュ関数は高速に計算できることが求められるため, 上記の 2 つの研究課題に加えて, IFRO ハッシュ関数の実装性能を改良することも重要である. 特に Sponge 構造は重要なハッシュ関数である SHA-3 の構造に採用さ

れており, Sponge の高速化は特に重要である. 本研究では 5 章で Sponge 構造の高速化を議論する.

2.9.4 研究課題 4

また, 速度に加えてハッシュ関数に求められる実装に関する要件として, ハッシュ関数のプログラムサイズや回路サイズは小さいことが好ましい. プログラムサイズや回路サイズを小さくすることができる定義域拡張構造として，ブロック暗号（例えば，AES）をプリミティブとする倍ブロック長定構造がある [20, 42, 34, 37, 47, 38, 39]. ブロック暗号とハッシュ関数を両方実装する場合，ブロック暗号を共通化して使うことができるなら，実装サイズを小さくすることができる. これまで, IFRO 安全性より弱い安全性である衝突困難性を満たす倍ブロック長構造 [34, 37, 47] は提案されているものの, IFRO 安全な倍ブロック長構造は提案されていない. 本研究では 6 章で IFRO 安全な倍ブロック長構造の設計について議論する.

2.9.5 研究課題 5

これまでの議論では, 定義域拡張構造 H の構成法について議論してきた. 一方で, ハッシュ関数を構成する場合は P の設計も必要である. IFRO 安全なハッシュ関数 H^{P} のプリミティブ P は, 例えば, 圧縮関数の場合はランダムオラクル圧縮関数, 置換の場合はランダム置換で理想化されているため, H^{P} を実装するうえで理想化された部品の代わりに用いる関数の設計も重要なテーマである.

理想的な部品の代わりとして用いる関数が満たすべき性質として現在有力となっているのが, Known-Key Distinguisher に対する安全性である [36]. これは, 理想的な部品 P の出力値には無視できる確率を除いて起こらない事象は設計した部品 P' でも無視できる確率を除いて起こらないことを考慮に入れた安全性である. この事象の例として, P は多くのビットが 0 となっている値を出力しない性質を持つこと, P の出力は無視できる確率を除いて衝突が起きないなどが挙げられる.

一方で, 設計した P' が Known-Key Distinguisher に対する安全性を担保していることを証明する方法は今のところ知られておらず, 現状は, 既存の攻撃法を P' に適用して, 適用できなければ P' は安全であると結論付けている. ここで, 最も強力な攻撃法の 1 つに差分攻撃があり [14, 15, 22, 58, 59, 61, 62, 60], P' の設計手法を確立するうえで差分攻撃の限界点を見極めることは重要な研究課題である. 本研究では 7 章で SHA-0, SHA-1 に対する差分攻撃の改良方法について議論する.

第 3 章

Merkle-Damgård 構造の救済

3.1 はじめに

$\mathcal{RO}$ 証明法（ランダムオラクル証明法）は暗号アルゴリズム $\mathcal{C}$ にハッシュ関数 H^{P} を用いた暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を保証する最も重要な証明技法である [7]．暗号システム $\mathcal{C}^{\mathsf{H}^{\mathsf{P}}}$ の $\mathcal{G}$-安全性を証明する場合，ハッシュ関数 H^{P} を理想化した $\mathcal{RO}$（ランダムオラクル）を用いる暗号システム $\mathcal{C}^{\mathcal{RO}}$ の $\mathcal{G}$-安全性を証明する．

これまで, この証明法を用いて多くの重要な暗号アルゴリズム (例えば, OAEP 暗号 [8], OAEP 暗号の改良方式 [19, 48, 49, 1], RSA-KEM [52]) の安全性が証明されている [7]. これらを暗号システムとして実装する際に, SHA-256 や SHA-512 といった FIPS で標準化されているハッシュ関数が最も用いられる [46]. これらのハッシュ関数は Merkle-Damgård (MD) 構造 [25, 41] を用いて構成されたハッシュ関数である．MD 構造は圧縮関数 $h : \{0,1\}^{r+n} \to \{0,1\}^n$ を繰り返す構造をもち, $\mathsf{MD}^h(m)$ の値は, 入力値 m に対して, パディング関数 pad を用いて m から r bit の倍数となる値 m^* を生成して, m^* を r bit ごとに分割した値を $m_1, \ldots, m_i$ として, $\mathsf{MD}^h(m) := h(h(\cdots h(IV, m_1), \cdots), m_i)$ とする．

しかし, $\mathcal{C}^{\mathcal{RO}}$ は安全であるが $\mathcal{C}^{\mathsf{MD}^h}$ は安全ではない暗号アルゴリズム $\mathcal{C}$ が知られており, MD 構造を用いたハッシュ関数は $\mathcal{RO}$ からの置き換えを保証できない．具体例として, ハッシュ関数を H^{P} として秘密鍵 K, メッセージ m に対して, M タグを $\mathsf{H}^{\mathsf{P}}(K\|m)$ の出力値とするメッセージ認証アルゴリズム Secret-Prefix MAC が知られている．$\mathsf{H}^{\mathsf{P}} = \mathsf{MD}^h$ の場合, MD 構造の h を繰り返す構造により, メッセージ m に対するタグ $tag = \mathsf{MD}^h(K\|m)$ から m の後ろに値 m^* を付加した新たなメッセージ $m\|m^*$ に対するタグ tag^* を K を知らずに tag と m^* から h を計算することで得ることができてタグの偽造ができるが, $\mathsf{H}^{\mathsf{P}} = \mathcal{RO}$ の場合, $\mathcal{RO}$ は一枚岩のランダム関数なのでこのような攻撃は適用できず, Secret Prefix MAC は $\mathcal{RO}$ モ

デルで安全性が証明できる. この攻撃法はメッセージ長を伸ばす攻撃法なのでメッセージ長拡張攻撃と呼ばれている.

このように, MD 構造を用いたハッシュ関数は Secret Prefix MAC の結果により, $\mathcal{RO}$ からの置き換えを保証できない. この問題に対して, Coron らは, Maurer らの Indifferentiability の枠組み [40] を用いて $\mathcal{RO}$ からの置き換えを保証できるハッシュ関数の安全性である Indifferentiability from a $\mathcal{RO}$ (IFRO) の安全性を考えて, ChopMD 構造など IFRO 安全となる構造をいくつか提案した [24]. そして, IFRO 安全性が提案されて以降, 多くのハッシュ関数がこの安全性を満たすように設計されている.

MD 構造に関する研究課題 Secret Prefix MAC の例によって, MD 構造を持つハッシュ関数は $\mathcal{RO}$ として使うことはできないが, MD 構造は FIPS の標準ハッシュ関数である SHA-256 や SHA-512 に採用されており, これからも多くの暗号システムで MD 構造を用いたハッシュ関数が用いられる. そして, $\mathcal{RO}$ モデルで安全性証明がつけられた OAEP 暗号やその改良方式, RSA-KEM, FDH 署名など実用的な暗号アルゴリズムに MD 構造を用いたハッシュ関数を用いても Secret Prefix MAC と同様の結果が成り立つとは限らない. すなわち, 以下のことを示すことは重要な課題である.

- OAEP 暗号やその改良方式, RSA-KEM, FDH 署名など実用的な暗号アルゴリズム $\mathcal{C}$ に MD 構造を持つハッシュ関数 MD^h を組み込んだ $\mathcal{C}^{\mathsf{MD}^h}$ が安全か否かを明らかにする.

そして, これらの暗号システムは今後多く実装されることが予想されるため, $\mathcal{C}^{\mathsf{MD}^h}$ は安全となることが望ましい.

3.1.1 成果.

本研究では, $\mathcal{RO}$ を組み込んだ時に安全な FDH 署名 [7], OAEP 暗号 [8], OAEP 暗号の改良方式 (例えば, SAEP 暗号 [19], 3R-OAEP 暗号 [48, 49], OAEP-4X 暗号 [1])), RSA-KEM [52] が MD 構造を持つハッシュ関数 MD^h を用いた暗号システムが安全であることを示す. ここで, h をランダムオラクル圧縮関数とする.

3.1.1.1 $\mathcal{WRO}$ 証明法

まず, これらの暗号システムの安全性を証明する方法論である $\mathcal{WRO}$ 証明法を提案する. MD^h は $\mathcal{RO}$ からの置き換えができないため, この方法論では, MD^h への置き換えができるよ

うに $\mathcal{RO}$ を弱めた Weakened $\mathcal{RO}$ ($\mathcal{WRO}$) を考えて, $\mathcal{WRO}$ から MD^h への置き換えを考える. 具体的な, $\mathcal{WRO}$ 証明法の証明手順は以下の通りである.

1. MD^h と $\mathcal{RO}$ の差はメッセージ拡張攻撃の適用可否なので, まず, メッセージ長拡張攻撃が適用できるように $\mathcal{WRO}$ を定義する.
2. そして, この $\mathcal{WRO}$ に対して, $\mathsf{MD}^h \sqsubset \mathcal{WRO}$ (Indifferentiable from $\mathcal{WRO}$ または IFWRO 安全) となることを示す.
3. 最後に, 暗号アルゴリズム $\mathcal{C}$ に対して, $\mathcal{WRO}$ を用いた暗号システム $\mathcal{C}^{\mathcal{WRO}}$ の安全性を証明する.

そして, 上記のことが証明できると, Indifferentiability の枠組みから, $\mathcal{C}^{\mathsf{MD}^h}$ は安全となる.

3.1.1.2 $\mathcal{WRO}$ 証明法を用いた $\mathcal{C}^{\mathsf{MD}^h}$ の安全性証明

$\mathcal{LRO}$ による MD^h を用いた FDH 署名の安全性. まず, $\mathcal{WRO}$ として Yoneyama らが提案した, $\mathcal{RO}$ と $\mathcal{RO}$ の入出力履歴を全て漏らすオラクル $\mathcal{LO}$ から構成される Leakey $\mathcal{RO}$ ($\mathcal{LRO}$) [63] を考える. 彼らは, FDH 署名に $\mathcal{WRO}$ を組み込んだ暗号システムが安全であることを証明した. ここで, $z = \mathcal{RO}(m)$ となる z を知っていたとすると, $\mathcal{LO}$ より m を得ることができるため, 任意の値 m' に対して $z = \mathcal{RO}(m\|m')$ を計算することができる. すなわち, メッセージ長拡張攻撃は $\mathcal{LRO}$ に適用可能であり, また, $\mathsf{MD}^h \sqsubset \mathcal{LRO}$ となることが証明できる. $\mathcal{WRO}$ 証明法から, FDH 署名に MD^h を用いた暗号システムは安全となる.

$\mathcal{TRO}$ による MD^h を用いた OAEP 暗号と OAEP 暗号の改良方式の安全性. $\mathcal{WRO}$ 証明法を用いて OAEP 暗号と OAEP 暗号の改良方式の安全性を証明する.

まず, $\mathcal{WRO} = \mathcal{LRO}$ とした時, OAEP 暗号と OAEP 暗号の改良方式は平文をハッシュ関数に入力するステップがあるため, $\mathcal{LO}$ を用いることで平文の情報を入手できて, これらの暗号に $\mathcal{LRO}$ を用いた暗号システムは安全ではないことが示せる.

そこで, メッセージ長拡張攻撃は適用できるが $\mathcal{RO}$ の全ての入出力値は漏らさずかつ, これらの方式で用いても安全となる $\mathcal{WRO}$ を考える. ここで, メッセージ長拡張攻撃は FDH 署名の部分で説明したように, $\mathcal{RO}$ の出力値からその入力値が得られるようなオラクルがあればよい. そして, OAEP 暗号と OAEP 暗号の改良方式の仕様としてこれらの内部で用いられるハッシュ関数の出力値は暗号文から漏れないようになっているため, $z = \mathcal{RO}(m)$ の z を入力として m のみを返信するオラクルを追加してもこれらの暗号アルゴリズムの安全性は担保できる. ここでは, このオラクルを Traceable Oracle ($\mathcal{TO}$) と呼ぶことにする. 以上の議論から,

$\mathcal{RO}$ と $\mathcal{TO}$ から構成される Traceable $\mathcal{RO}$ ($\mathcal{TRO}$) を $\mathcal{WRO}$ とする.

そして, $\mathcal{RO}(m)$ の出力値 z と別の値 m' が与えら得ている場合, z から m を得ることができるので, $\mathcal{RO}$ に $m\|m'$ をクエリーすることでその出力値 z' を得ることができ, z と m' から $m\|m'$ の出力値 z' を得ることができるメッセージ長拡張攻撃に成功する. また, $\mathsf{MD}^h \sqsubset \mathcal{TRO}$ となることを証明できる. さらに, OAEP 暗号と OAEP 暗号の改良方式に $\mathcal{TRO}$ を組み込んだ暗号システムが安全であることが証明できる. $\mathcal{WRO}$ 証明法から, OAEP 暗号と OAEP 暗号の改良方式に MD^h を用いた暗号システムは安全となる.

$\mathcal{ERO}$ による MD^h を用いた RSA-KEM の安全性. 次に, $\mathcal{WRO}$ 証明法を用いて RSA-KEM の安全性を証明する.

まず, RSA-KEM に $\mathcal{TRO}$ を用いた暗号システムの場合, RSA-KEM の暗号文から RSA-KEM 内部で用いられるハッシュ関数の出力値が分かるため, $\mathcal{TO}$ にこのハッシュ値をクエリするとその入力値を得ることができて, RSA-KEM に $\mathcal{TRO}$ を用いた暗号システムが安全ではないことが示せる.

そこで, メッセージ長拡張攻撃は適用できるが $\mathcal{RO}$ の全ての入出力値は漏らさずかつ, これらの方式で用いても安全となる $\mathcal{WRO}$ を考える. これまでの $\mathcal{WRO}$ は, $z = \mathcal{RO}(m)$ となる z に対して, サブオラクルを用いて m を入手して, m とある m' に対して $z' = \mathcal{RO}(m\|m')$ により z' を計算していたが, $\mathcal{RO}$ の出力値から入力値を返すオラクルがある場合, RSA-KEM の安全性が証明できないため, z と m' から $z' = \mathcal{RO}(m\|m')$ となる z' を返すオラクルである Extention Oracle ($\mathcal{EO}$) を定義する. そして, $\mathcal{WRO}$ を $\mathcal{RO}$ と $\mathcal{EO}$ から構成されるオラクルとして定義する. このオラクルを Extension Attack Simulatable $\mathcal{RO}$ ($\mathcal{ERO}$) と呼ぶことにする. $\mathcal{EO}$ によりメッセージ長拡張攻撃が適用できて, $\mathsf{MD}^h \sqsubset \mathcal{ERO}$ となることが証明できる. そして, $\mathcal{ERO}$ からハッシュ関数の入力値は漏れないので, RSA-KEM に $\mathcal{ERO}$ を用いた暗号システムの安全性が証明できる. $\mathcal{WRO}$ 証明法から, RSA-KEM に MD^h を用いた暗号システムは安全となる.

MD^h と $\mathcal{ERO}$ の等価性. $\mathsf{MD}^h \sqsubset \mathcal{ERO}$ となるが, さらに, $\mathcal{ERO} \sqsubset \mathsf{MD}^h$ となることを示す. すなわち, MD^h を用いた暗号システムが安全ならば, MD^h の代わりに $\mathcal{ERO}$ を用いても安全となる. また, その逆も成り立つ. よって, MD^h と $\mathcal{ERO}$ は等価であるため, $\mathcal{ERO}$ を用いた暗号システムの安全性を評価すれば, MD^h を用いた暗号システムの安全性を評価できる.

Separations. RSA-KEM に $\mathcal{ERO}$ を組みこんだ暗号システムは安全であるものの, $\mathcal{TRO}$ を組み込んだ時に安全ではなくなるため, RSA-KEM が separation となり, $\mathcal{TRO}$ は $\mathcal{ERO}$ より

真に弱いオラクルと言える.

Yoneyama らは OAEP 暗号に $\mathcal{LRO}$ を組み込んだ暗号システムが安全ではないことを示した. OAEP 暗号は $\mathcal{TRO}$ を組み込んだ時に安全となるため, OAEP 暗号が separation となり, $\mathcal{LRO}$ は $\mathcal{TRO}$ より真に弱いオラクルと言える.

最後に, $\mathcal{RO}$ と $\mathcal{ERO}$ の separation を考える. prefix-MAC [57] は $\mathcal{RO}$ を組み込んだときに安全であることが知られているものの, メッセージ長拡張攻撃が成功すると安全でなくなることが知られている. すなわち, prefix-MAC は $\mathcal{ERO}$ を組み込んだときには安全ではない. よって, $\mathcal{ERO}$ は $\mathcal{RO}$ より真に弱いオラクルである.

$\mathcal{WRO}$ 証明法 vs. ダイレクト証明法. ここで, 暗号アルゴリズム $\mathcal{C}$ に MD^h を用いた暗号システム $\mathcal{C}^{\mathsf{MD}^h}$ の安全性は, $\mathcal{WRO}$ 証明法を用いずに, ダイレクトに $\mathcal{C}^{\mathsf{MD}^h}$ の安全性を証明することで証明できるものの, ダイレクト証明を考える場合, 暗号アルゴリズムの構造と MD 構造を同時に考えて証明するため, 証明が複雑になる. 一方, $\mathcal{WRO}$ 証明法を用いることで, 暗号システムの安全性を評価するときに, MD 構造を考えずに証明できるため, $\mathcal{WRO}$ 証明を用いた方がダイレクト証明法より証明が容易になるメリットがある.

関連文献. Dodis たちは, 本研究とは独立に, public-use random oracle (pub-RO と書く) を用いて MD 構造を救済した [28].

彼らは, ハッシュ関数の出力が全て公開で, $\mathcal{RO}$ を組み込んだ時に安全な任意の暗号アルゴリズムは pub-RO を組み込んでも安全となることを示した. すなわち, FDH 署名, PSS 署名, Fiat-Shamir 署名など多くの暗号アルゴリズムが pub-RO を組み込んでも安全となる. そして, MDHF が indifferentiable from pub-RO となることを示した. すなわち, これらの暗号アルゴリズムに MDHF を組み込んだ暗号システムは安全となる.

ここで, pub-RO は Yoneyama らが提案した $\mathcal{LRO}$ と同じであるため, 彼らの成果は OAEP 暗号とその改良方式, そして RSA-KEM に MD^h を用いた暗号システムを救済できない.

結論. 以上より, FDH 署名, PSS 署名, Fiat-Shamir 署名などの暗号アルゴリズムに MD^h を用いた暗号システムは我々の成果と Dodis たちの成果を併せると救うことができ, OAEP 暗号やその改良暗号, そして RSA-KEM に MD^h を組み込んだ暗号システムは我々の成果から救うことができた. よって, MD^h を組み込んだ多くの暗号システムは安全であるため, MD 構造は実用的には問題なく使用できると結論付けることができる.

3.2 $\mathcal{WRO}$ の定義と MD 構造への適用

本章では, メッセージ長拡張攻撃を適用可能な弱めた $\mathcal{RO}$ ($\mathcal{WRO}$ と書く) を定義し, $\mathcal{WRO}$ を用いて MD^h is indifferentiable from $\mathcal{WRO}$ となることを証明する.

3.2.1 $\mathcal{WRO}$ の定義

ここでは, $\mathcal{WRO}$ として, Leaky $\mathcal{RO}$ ($\mathcal{LRO}$ と書く), Traceable $\mathcal{RO}$ ($\mathcal{TRO}$ と書く), Extension Attack Simulatable $\mathcal{RO}$ ($\mathcal{ERO}$ と書く) を定義する.

3.2.2 $\mathcal{LRO}$

$\mathcal{LRO}$ は Yoneyama らによって提案された弱めた $\mathcal{RO}$ である [63]. $\mathcal{LRO}$ は $\mathcal{RO}$ と Leaky オラクル $\mathcal{LO}$ から構成される. $\mathcal{LO}$ は $\mathcal{RO}$ のハッシュリストを返すオラクルである.

3.2.2.1 $\mathcal{TRO}$

$\mathcal{TRO}$ は $\mathcal{RO}$ と Trace Oracle ($\mathcal{TO}$ と書く) から構成される. $\mathcal{TO}$ の定義は以下の通りである. $\mathcal{TO}$ へのクエリは n bit の値 z で, z に対して以下の動作を行う.

1. If $\exists(m_1, z), (m_2, z), \ldots \in \mathcal{L}$ then returns these pairs.
2. Else returns $\perp$.

3.2.2.2 $\mathcal{ERO}$

$\mathcal{ERO}$ は $\mathcal{RO}$ と Extension Attack Oracle ($\mathcal{EO}$ と書く) から構成される. $\mathcal{EO}$ の定義は以下の通りである. $\mathcal{EO}$ へのクエリは t bit の値 m' と n bit の値 z で, (m', z) に対して以下の動作を行う. ここで, $\mathcal{L}_{\mathcal{EO}}$ を $\emptyset$ で初期化されたテーブルとする.

1. If $(m', z, z') \in \mathcal{L}_{\mathcal{EO}}$ then returns z'.
2. Else if $z = IV$ then $\mathcal{EO}$ poses query m to $\mathcal{RO}$, receives z', $\mathcal{L}_{\mathcal{EO}} \xleftarrow{\cup} (m, z, z')$, and returns z'.
3. Else if there exists only one pair $(m, z) \in \mathcal{L}$, $\mathcal{EO}$ poses query $m||m'$ to $\mathcal{RO}$, receives z', $\mathcal{L}_{\mathcal{EO}} \leftarrow (m', z, z')$, and returns z'.
4. Else $z' \xleftarrow{\$} \{0,1\}^n$, $\mathcal{L}_{\mathcal{EO}} \xleftarrow{\cup} (m, z, z')$ and returns z'.

3.2.3 $\mathsf{MD}^h \sqsubset \mathcal{ERO}$ と $\mathcal{ERO} \sqsubset \mathsf{MD}^h$

本章で, MD^h と $\mathcal{ERO}$ が等しいことを証明する. すなわち, $\mathsf{MD}^h \sqsubset \mathcal{ERO}$ と $\mathcal{ERO} \sqsubset \mathsf{MD}^h$ を証明する.

定理 3.1 S をあるシミュレータとする. そして, 任意の識別者 $\mathcal{D}$ に対して, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MD}^h,\mathcal{ERO},S}(\mathcal{D}) \leq \frac{(\sigma_H + q_h)(2(\sigma_H + q_h) + 1)}{2^n}.$$

ここで, σ_H は $\mathcal{D}$ から $\mathcal{RO}/\mathsf{MD}^h$ へのクエリのトータルの (r-bit) メッセージブロック数で q_h は $\mathcal{D}$ の S/h へのクエリ回数である. そして, S の動作時間は $t_S = O(q_h)$ である.

この証明は 3.2.5 章で行う.

定理 3.2 S をあるシミュレータとする. そして, 任意の識別者 $\mathcal{D}$ に対して, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathcal{ERO},\mathsf{MD}^h,S}(\mathcal{D}) \leq \frac{(\sigma_H + q_{\mathcal{EO}})(2(\sigma_H + q_{\mathcal{EO}}) + 1)}{2^n}.$$

ここで, σ_H は $\mathcal{D}$ から $\mathcal{RO}/\mathsf{MD}^h$ へのクエリのトータルの (r-bit) メッセージブロック数で $q_{\mathcal{EO}}$ は $\mathcal{D}$ の $\mathcal{EO}/S$ へのクエリ回数である. そして, S の動作時間は $t_S = O(q_{\mathcal{EO}})$ である.

この証明は 3.2.6 章で行う.

3.2.4 $\mathsf{MD}^h \sqsubset \mathcal{TRO}$

本章で, $\mathsf{MD}^h \sqsubset \mathcal{TRO}$ を証明する.

定理 3.1 より $\mathsf{MD}^h \sqsubset \mathcal{ERO}$ となるので, $\mathcal{ERO} \sqsubset \mathcal{TRO}$ ならば $\mathsf{MD}^h \sqsubset \mathcal{TRO}$ となる. 以下, $\mathcal{ERO} \sqsubset \mathcal{TRO}$ を証明する.

定理 3.3 S をあるシミュレータとする. そして, 任意の識別者 $\mathcal{D}$ に対して, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathcal{ERO},\mathcal{TRO},S}(\mathcal{D}) = 0.$$

ここで, S の動作時間は $t_S = O(q^2)$ である.

証明. リスト L を $\emptyset$ で初期化して, S を以下のように定義する.

Simulator $S(m', z)$
01 If $(m', z, z') \in L$, returns z'
02 $T \leftarrow \mathcal{TO}(z)$.
03 If $z = IV$ then $z' \leftarrow \mathcal{RO}(m')$, $L \xleftarrow{\cup} (m', z, z')$ and returns z'.
04 Else if only one value m is stored in T, $z' \leftarrow \mathcal{RO}(m||m')$, $L \leftarrow (m', z, z')$ and returns z'.
05 Else, $z' \xleftarrow{\$} \{0,1\}^n$, $L \leftarrow (m', z, z')$ and returns z'.

上記の手続きは $\mathcal{EO}$ の手続きと同じ手続きである. よって, $(\mathcal{RO}, S)$ と $(\mathcal{RO}, \mathcal{EO})$ は区別できないため, 上記の定理が成り立つ.

■

3.2.5 定理 3.1 の証明

まず, Chain Triple を定義する.

定義 16 (Chain Triples)
$x_1 = IV$ かつ $y_j = x_{j+1}$ $(j = 1, \ldots, j-1)$ となる $(m_1, x_1, y_1), \ldots, (m_i, x_i, y_i)$ を chain triples と呼ぶことにする.

ここで, Chain Triples は MD 構造の繰り返し構造から得られる入出力値を考えた構造となっていることに注意されたい.

次に, シミュレータを定義する.

Simulator $S(x, m)$
$y \leftarrow \mathcal{EO}(m, x)$.
returns y.

このシミュレータの動作時間は $O(q_h)$ である.

そして, 本証明では, 以下の 3 つのゲームを考えて証明を行う. ここで, $\mathcal{D}$ は (L, R) にアクセスできるとする.

- **Game 0**: このゲームは Ideal World である. すなわち, $(L, R) = (\mathcal{RO}, S)$ である.
- **Game 1**: このゲームで, L を $\mathcal{RO}$ から MD^S に変更する. すなわち, $(L, R) =$

(MD^S, S) である．ここで，MD^S は MD 構造に S を圧縮関数として組み込んだハッシュ関数である．

- **Game 2**: このゲームは最終ゲームで，Real World を考える．すなわち，$(L, R) = (\mathsf{MD}^h, h)$ である．

ここで，Gi を Game i で D が 1 を出力するイベントとする．よって，$\Pr[G0] = \Pr[D^{\mathcal{RO},S} \Rightarrow 1]$，$\Pr[G2] = \Pr[D^{\mathsf{MD}^h,h} \Rightarrow 1]$ となる．

Game 0 ⇒ Game 1: 以下，$\Pr[G1] - \Pr[G0]$ を評価する．まず，$\mathcal{D}$ から $\mathcal{RO}$ への初出のクエリ m に対する出力 z に注目した以下の bad イベントを考える．

- Event $\mathsf{E1}$: $m \neq m'$ かつ $z = z'$ となるペア (m', z') が $\mathcal{L}_{\mathcal{RO}}$ に記録されている．
- Event $\mathsf{E2}$: $z = x'$ となる 3 つ組 (m', x', y') が $\mathcal{L}_{\mathcal{EO}}$ に記録されている．
- Event $\mathsf{E3}$: $z = IV$．

次に，以下のことを証明する．

1. Game 1 で，上記のイベントが起こらなければ，L へのクエリ m のレスポンスは $\mathcal{RO}(M)$ となる．
2. Game 0 で，上記のイベントが起こらなければ，Game 1 で保証されている R のレスポンスは L のレスポンスとの関係を保証する．すなわち，Game 0 で，S によって定義される ($\mathcal{L}_{\mathcal{EO}}$ 内の) 任意の chain triples $(m_1, x_1, y_1), \ldots, (m_i, x_i, y_i)$ に対して，$y_i = \mathcal{RO}(m_1 || \cdots || m_i)$ となる．

Game 0 と Game 1 の違いは L が R を用いているか否かであり，上記のことが示せると，3 つのイベントが起こらない限り，Game 0 と Game 1 は等しいと結論付けることができる．すなわち，以下の式が成り立つ．

$$\Pr[G1] - \Pr[G0] \leq \Pr[\mathsf{E1} \vee \mathsf{E2} \vee \mathsf{E3}]$$

まず，$\Pr[\mathsf{E1} \vee \mathsf{E2} \vee \mathsf{E3}]$ を評価するための第 1 歩として以下の補助命題を証明する．

補題 4 $\mathcal{L}_{\mathcal{EO}}$ 内の任意の chain triples (m_1, x_1, y_1), …, (m_i, x_i, y_i) に対して，bad イベントが起こらないならば，$y_i = \mathcal{RO}(m_1 || ... || m_i)$ となる．

証明. 補助命題の対偶をとると，$\mathcal{L}_{\mathcal{EO}}$ 内のある chain triples $(m_1, x_1, y_1), \ldots, (m_i, x_i, y_i)$ に対

して, $y_i \neq \mathcal{RO}(m_1||...||m_i)$ ならば, bad イベントが起こる. 以下, chain triples (m_1, x_1, y_1), $\ldots$, (m_i, x_i, y_i) に対して, $y_i \neq \mathcal{RO}(m_1||...||m_i)$ となることを仮定して, bad イベントが起こることを証明する.

まず, 以下の 2 ケースに分ける.

- Case 1: $\forall j \in \{1, \ldots, i\} : y_j \neq \mathcal{RO}(m_1||\cdots||m_j)$.
- Case 2: $\exists j \in \{1, \ldots, i-1\}$ s.t. $y_j = \mathcal{RO}(m_1||\cdots||m_j)$. (ここで, $y_i \neq \mathcal{RO}(m_1||\cdots||m_i)$ なので, $j \neq i$ となることに注意されたい)

Case 1. $x_1 = IV$ なので, $\mathcal{EO}$ の仕様から $y_1 = \mathcal{RO}(m_1)$ となる. よって, このケースは起きない.

Case 2. j を $y_j = \mathcal{RO}(m_1||\cdots||m_j)$ となる $\{1, \ldots, i-1\}$ 内の最大値とする. 次に以下の 2 ケースに分ける.

- Case 2-1: $(m_{j+1}, x_{j+1}, y_{j+1})$ が $\mathcal{RO}$ によって定義される.
- Case 2-2: $(m_{j+1}, x_{j+1}, y_{j+1})$ は $\mathcal{RO}$ で定義されない.

まず, Case 2-1 では, $y_{j+1} = \mathcal{RO}(m||m_{j+1})$ となる m が存在する. そして, j の条件より, $m \neq m_1||\cdots||m_j$ となる.

次に, Case 2-1 を (Case 2-1-1) $m = \perp$ と (Case 2-1-2) $m \neq \perp$ に分ける.

Case 2-1-1 では, $y_{j+1} = \mathcal{RO}(m_{j+1})$ が成り立つ. そして, $\mathcal{ERO}$ の定義より, $(m_{j+1}, x_{j+1}, y_{j+1})$ は $\mathcal{EO}$ の Step 2 で定義される. $x_{j+1} = y_j = \mathcal{RO}(m_1||\cdots||m_j)$ かつ $x_{j+1} = IV$ なので, イベント E3 が起こる.

Case 2-1-2 では, $\mathcal{ERO}$ の定義から, $(m_{j+1}, x_{j+1}, y_{j+1})$ は $\mathcal{EO}$ の Step 2 または Step 3 で定義される. $x_{j+1} = y_j = \mathcal{RO}(m_1||\cdots||m_j)$ かつ $x_{j+1} = \mathcal{RO}(M)$ なので, イベント E1 が起こる.

最後に Case 2-2 を考える. $(m_{j+1}, x_{j+1}, y_{j+1})$ は $\mathcal{RO}$ で定義されないので, この 3 つ組は $\mathcal{EO}$ の Step 4 で定義される. $x_{j+1} = y_j = \mathcal{RO}(m_1||\cdots||m_j)$ が成り立つため, (m_j, x_j, y_j) が定義された時には, $(m_{j+1}, x_{j+1}, y_{j+1})$ はすでに定義されていたことになる. よって, このケースではイベント E2 が起こる.

以上より, $y_i \neq \mathcal{RO}(m_1||\cdots||m_i)$ ならば bad イベントが起こることが示せた.

■

次に, bad イベントが起こる確率を以下で評価する.

補題 5 q_1 を S の最大呼び出し回数, q_2 を $\mathcal{RO}$ の最大呼び出し回数とする. この場合, 以下の式が成り立つ.

$$\Pr[\mathsf{E1} \vee \mathsf{E2} \vee \mathsf{E3}] \leq \frac{q_2^2 + q_1 q_2 + q_2}{2^n}$$

証明. E1 は $\mathcal{RO}$ のコリジョンイベントなので誕生日解析 (詳しくは付録を参照されたい) を適用すると, 以下の式が成り立つ.

$$\Pr[\mathsf{E1}] \leq \frac{q_2^2}{2^n}.$$

E2 は乱数値がある固定値にぶつかるイベントなので, 以下の式が成り立つ.

$$\Pr[\mathsf{E2}] \leq \frac{q_1 q_2}{2^n}.$$

E3 は乱数値が IV と一致するイベントなので, 以下の式が成り立つ.

$$\Pr[\mathsf{E3}] \leq \frac{q_2}{2^n}.$$

よって, 以下の式が成り立つ.

$$\Pr[\mathsf{E1} \vee \mathsf{E2} \vee \mathsf{E3}] \leq \Pr[\mathsf{E1}] + \Pr[\mathsf{E2}] + \Pr[\mathsf{E3}] \leq \frac{q_2^2 + q_1 q_2 + q_2}{2^n}.$$

■

B0 を, Game 0 での bad イベントとする. B1 を, Game 1 での bad イベントとする.

Game 0 では, S は $\mathcal{D}$ のみから呼び出されるので, $q_1 = q_h$ となる. また, $\mathcal{RO}$ は $\mathcal{D}$ と $\mathcal{EO}$ から呼び出されるので, $q_1 = \sigma_H + q_h$ となる.

Game 1 では, S は L と $\mathcal{D}$ から呼び出されるため, $q_1 = \sigma_H + q_h$ となる. $\mathcal{RO}$ は S から呼び出されるため, $q_2 = \sigma_H + q_h$ となる. Lemma 5 より, 以下の確率が成り立つ.

$$\Pr[\mathsf{B0}] \leq \frac{(\sigma_H + q_h)(\sigma_H + 2q_h + 2)}{2^n}, \ \Pr[\mathsf{B1}] \leq \frac{(\sigma_H + q_h)(2(\sigma_H + q_h) + 1)}{2^n}$$

以上より, 以下の式が成り立つ.

$$\Pr[G1] - \Pr[G0] \leq \max\{Pr[\mathsf{E0}], Pr[\mathsf{E1}]\} = \frac{(\sigma_H + q_h)(2(\sigma_H + q_h) + 1)}{2^n}$$

Game 1 ⇒ Game 2: 最後に, $\Pr[G2] - \Pr[G1]$ を評価する. S の出力は $\mathcal{EO}$ によってランダムに選ばれるため, S と h の区別はつかない. よって, $Pr[G2] = Pr[G1]$ となる.

以上で証明終わりである.

■

3.2.6 定理 3.2 の証明

シミュレータ S を以下に定義する.

$$\underline{\textbf{Simulator } S(m, z)}$$
$$z' \leftarrow h(z, m).$$
$$\text{returns } z'.$$

このシミュレータの動作時間は $O(q_{\mathcal{EO}})$ である.

この証明では定理 3.1 を利用する. そして, 以下の 3 つのゲームを考える. ここで, 各々のゲームで $\mathcal{D}$ は (L, R) と対話する.

- **Game 0**: このゲームは Real World である. すなわち, $(L, R) = (\mathcal{RO}, \mathcal{EO})$.
- **Game 1**: このゲームでは $(L, R) = (\mathsf{MD}^h, h)$ とする.
- **Game 2**: このゲームでは R を h から S に変更する. すなわち, $(\mathcal{O}_H, \mathcal{O}_{\mathcal{EO}}) = (\mathsf{MD}^h, S)$.

Gi を Game i で $\mathcal{D}$ が 1 を出力するイベントとする. よって, $Pr[G0] = Pr[\mathcal{D}^{\mathcal{ERO}} \Rightarrow 1]$, $Pr[G2] = Pr[\mathcal{D}^{\mathsf{MD}^h, S} \Rightarrow 1]$ となる.

Game 0 ⇒ Game 1. 定理 3.1 のシミュレータは $\mathcal{EO}$ そのものであった. よって, この差の評価で定理 3.1 の結果を適用可能で, 以下の式が成り立つ.

$$|\Pr[G1] - \Pr[G0]| \leq \frac{(\sigma_H + q_{\mathcal{EO}})(2(\sigma_H + q_{\mathcal{EO}}) + 1)}{2^n}$$

Game 1 ⇒ Game2. $S = h$ なので, Game 1 と Game 2 は同じである. すなわち, $Pr[G2] = Pr[G1]$.

以上で証明終わりである.

3.3 $\mathcal{TRO}$ モデルでの OAEP 暗号の安全性

Optimal Asymmetric Encryption Padding (OAEP) 暗号 [8] は $\mathcal{RO}$ モデルで安全な公開鍵暗号アルゴリズムである. 本章では, $\mathcal{TRO}$ モデルで OAEP 暗号の安全性を証明する.

3.3.1 Asymmetric Encryption Schemes の安全性

まず, 公開鍵暗号の安全性を定義する.

定義 17 公開鍵暗号は以下の 3 つ組 $(\mathbf{EGen}, \mathbf{Enc}, \mathbf{Dec})$ から構成される:

> **EGen**： 鍵生成アルゴリズム. k をセキュリティパラメータとして, 1^k を入力として受け取り, 暗号化の鍵と復号の鍵のペア (ek, dk) を出力する.
> **Enc**： 暗号化アルゴリズム. 暗号化鍵 ek とメッセージ m を入力とし, 暗号文 c を出力する.
> **Dec**： 復号アルゴリズム. 復号鍵 dk と暗号文 c を受け取り, 平文 m またはエラー値 $\perp$ を出力する.

公開鍵暗号の安全性は一方向性や識別付不可能性といったいくつかの安全性で定義される. 一般的には, 選択暗号文攻撃での識別不可能性 (IND-CCA) は最も強い公開鍵暗号の安全性となっている. 以下, IND-CCA の安全性を説明する.

定義 18 (IND-CCA) セキュリティパラメータ k に対して以下の性質を満たすならば, その公開鍵暗号は (t, ϵ)-IND-CCA 安全する: 任意の攻撃者 $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ に対して, $|\Pr[(ek, dk) \leftarrow \mathbf{EGen}(1^k);\ (m_0, m_1, state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(dk,\cdot)}(ek);\ b \stackrel{R}{\leftarrow} \{0,1\};\ c^* \leftarrow \mathbf{Enc}(ek, m_b);\ b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(dk,\cdot)}(ek, c^*, state);\ b' = b] - 1/2| \leq \epsilon$. ここで, $\mathcal{DO}$ は復号オラクル, $state$ はステイト情報 (ek, m_0, m_1 を含む) で, $\mathcal{A}$ は t steps 動作するとする. $\mathcal{A}$ は $c = c^*$ を $\mathcal{DO}$ にクエリできない.

3.3.2 OAEP

OAEP 暗号は trapdoor partial-domain one-way permutation をベースとするアルゴリズムである. trapdoor partial-domain one-way permutation の定義は以下の通りである.

定義 19 $\mathcal{G}$ を a trapdoor permutation generator とする. 以下のことが成り立つ f を (t, ϵ)-partial-domain one-way と呼ぶことにする.

- 入力 1^k に対して, $\mathcal{G}$ は (f, f^{-1}, Dom) を出力する. ここで, Dom は $\{0,1\}^{k_0} \times \{0,1\}^{k_1}$ の部分集合で, $(k_0 + k_1 < k)$, f, f^{-1} は Dom 上の置換である.
- f が $p(k)$ の時間内で計算可能な多項式 p が存在する.
- 攻撃者 Alg に対して, $\Pr[(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k);\ (x_0, x_1) \stackrel{R}{\leftarrow} Dom;\ Alg(f,\ Dom,\ f(x_0, x_1)) = x_0] \leq \epsilon$ となる. ここで, Alg は多くても t ステップ動作するとする.

OAEP 暗号アルゴリズムの仕様は以下の通りである.

鍵生成アルゴリズム : セキュリティパラメータ k に対して, 暗号化鍵 $(ek = f)$ と復号鍵 $(dk = f^{-1})$ を出力する. ここで, $(f, f^{-1}, Dom = \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}) \leftarrow \mathcal{G}(1^k)$ である. $\mathcal{G}$ は trapdoor permutation generator で $n = k - k_0 - k_1$ である.

暗号化アルゴリズム : メッセージ $m \in \{0,1\}^n$ に対して, 乱数 $r \stackrel{R}{\leftarrow} \{0,1\}^{k_0}$ を生成し, $x = (m||0^{k_1}) \oplus G(r)$ と $y = r \oplus H(x)$ を計算し, 暗号文 $c = f(x, y)$ を出力する. $H : \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$ と $G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ はハッシュ関数である.

復号アルゴリズム : 暗号文 c に対して, $z = f^{-1}(c)$ を計算し, z を (x, y) に分割し, $r = y \oplus H(x)$ とする. ここで, $|x| = n + k_1$ and $|y| = k_0$ である.
そして, もし, $[x \oplus G(r)]_{k_1} \stackrel{?}{=} 0^{k_1}$ ならば, $m = [x \oplus G(r)]^n$ を c の平文として出力する.
$[a]^b$ は a の下位 b bit の値で, $[a]_b$ は a の上位 b bit の値である.
それ以外は, c を不適切な暗号文としてリジェクトする.

文献 [30] では, 以下の $\mathcal{RO}$ モデルの OAEP 暗号アルゴリズムの安全性が示されている;

補題 6 ([30]) もしトラップドア f が partial-domain one-way ならば, OAEP 暗号は, H と G が $\mathcal{RO}$ の場合に, IND-CCA 安全である.

3.3.3 OAEP 暗号アルゴリズムの $\mathcal{LRO}$ モデルでの不安全性

OAEP 暗号は $\mathcal{RO}$ モデルで IND-CCA 安全性が示されいるが, $\mathcal{LRO}$ モデルでは破られることが示されている. 具体的には, IND-CCA より弱い安全性である OW-CPA に対して, OAEP 暗号が $\mathcal{LRO}$ モデルで破られることが示されている.

補題 7 ([63]) trapdoor permutation f が partial-domain one-way でも, H と G が $\mathcal{LRO}$ ならば, OAEP 暗号は OW-CPA 安全性を満たさない.

3.3.4 $\mathcal{TRO}$ モデルでの OAEP 暗号の安全性

以下, $\mathcal{TRO}$ モデルでの OAEP 暗号の安全性を証明する.

定理 3.4 trapdoor permutation f が (t', ϵ')-partial-domain one-way ならば OAEP 暗号は以下の (t, ϵ)-IND-CCA を満たす:

$$
\begin{aligned}
t' &= t + q_{RG} \cdot q_{RH} \cdot perm, \\
\epsilon' &\geq \frac{1}{q_{RH}} \cdot \left(\frac{\epsilon}{2} - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}} \right) \\
&\quad - \frac{q_{TH}}{2^{k_0}},
\end{aligned}
$$

ここで, H と G は $\mathcal{TRO}$ であり, q_{RH} を H の $\mathcal{RO}$ へのクエリ回数, q_{TH} を H の $\mathcal{TO}$ へのクエリ回数, q_{RG} を G の $\mathcal{RO}$ へのクエリ回数, q_{TG} を G の $\mathcal{TO}$ へのクエリ回数とする. q_D を復号オラクル $\mathcal{DO}$ へのクエリ回数, $perm$ を f の動作時間とする.

3.6 章で証明を与える.

備考 5 定理 3.4 のように, 改良 OAEP 暗号に対しても $\mathcal{TRO}$ モデルで安全性を署名することができる. 改良 OAEP 暗号は, 例えば, OAEP+ [53], SAEP [19], OAEP-3R [49], OAEP-4X [1] があり, これらは $\mathcal{TRO}$ モデルで安全となる.

3.4 RSA-KEM の安全性解析

RSA ベースの key encapsulation mechanism (RSA-KEM) [52] は $\mathcal{RO}$ モデルで安全な KEM である. 本章では, RSA-KEM が $\mathcal{TRO}$ モデルと $\mathcal{ERO}$ モデルで安全となることを証明

する.

3.4.1 KEM の安全性定義

まず, KEM のモデルを説明する.

定義 20 KEM は以下の 3-tuple (**KEM**.**Gen**, **KEM**.**Enc**, **KEM**.**Dec**) から構成される:

> **KEM**.**Gen**： 鍵生成アルゴリズム. 鍵生成アルゴリズム. k をセキュリティパラメータとして, 1^k を入力として受け取り, 暗号化の鍵と復号の鍵のペア (ek, dk) を出力する.
> **KEM**.**Enc**： 暗号化アルゴリズム. 暗号化鍵 ek を入力とし, 鍵 K と暗号文 c を出力する.
> **KEM**.**Dec**： 復号アルゴリズム. 復号鍵 dk と暗号文 c を受け取り, 鍵 K またはエラー値 $\perp$ を出力する.

KEM の安全性は公開鍵暗号と同様に IND-CA で定義される. 以下, KEM に対する IND-CCA 安全性である.

定義 21 もしセキュリティパラメータ k に対して以下の性質を満たすならば, KEM アルゴリズムは (t, ϵ)-IND-CCA 安全とする; 任意の攻撃者 $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ に対して, $|\Pr[\ (ek, dk) \leftarrow \mathbf{KEM}.\mathbf{Gen}(1^k);\ (state) \leftarrow \mathcal{A}_1^{\mathcal{DO}(dk,\cdot)}(ek);\ b \xleftarrow{R} \{0,1\};\ (K_0^*, c_0^*) \leftarrow \mathbf{KEM}.\mathbf{Enc}(ek);\ K_1^* \xleftarrow{R} \mathcal{K};\ b' \leftarrow \mathcal{A}_2^{\mathcal{DO}(dk,\cdot)}(ek,\ (K_b^*, c_0^*),\ state);\ b' = b] - 1/2| \leq \epsilon$ となる. ここで, $\mathcal{DO}$ は復号オラクル, $\mathcal{K}$ は鍵空間, $state$ は $\mathcal{A}_1$ から $\mathcal{A}_2$ に引き継がれるステイト情報で, $\mathcal{A}$ の動作時間を t ステップとする. また, $\mathcal{A}$ は暗号文 $c = c_0^*$ を $\mathcal{DO}$ にクエリできない.

3.4.2 RSA-KEM

RSA-KEM は以下の RSA 仮定をベースとする方式である.

定義 22 n を RSA の法とする. n は 2 つの大きな素数 (p, q) で, セキュリティパラメータ k に対して $|p| = |q| = k$ で, e は $\gcd\ (e,\ \phi(n)) = 1$ となるべき指数である. ここで, $\phi(n) = (p-1)(q-1)$. もし, 任意の攻撃者 Alg に対して, $\Pr[y \leftarrow \mathbb{Z}_n;\ Alg(n, e, y) = x;\ y \equiv x^e \pmod{n}] \leq \epsilon$ ならば RSA 問題が (t, ϵ)-hard ということにする. ここで, Alg は t ステップで動作する攻撃者とする.

RSA-KEM の仕様は以下の通り:

鍵生成： 入力 k に対して, 暗号化鍵 $(ek = (n, e))$, 復号鍵 $(dk = d)$ を出力する. ここで, n は RSA で用いる法である. また, 安全性パラメータ k, $\gcd(e, \phi(n)) = 1$ and $ed \equiv 1 \pmod{\phi(n)}$ に対して, 大きな素数 (p, q) が生成される.

暗号化： 乱数 $r \xleftarrow{R} \mathbb{Z}_n$ を生成し, $c = r^e \mod n$ and $K = H(r)$ を計算し, 暗号文 c と鍵 K を生成する. ここで, $H : \mathbb{Z}_n \to \{0,1\}^k$ はハッシュ関数である.

復号： 暗号文 c に対して, $c \in \mathbb{Z}_n$ か否かをチェックする. もし正しくないならば, エラーシンボル $\perp$ を出力, 正しいならば, $r = c^d \mod n$ を計算し, $K = H(r)$ を出力する.

$\mathcal{RO}$ モデルでの RSA-KEM の安全性は以下の通りである.

補題 8 ([52]) もし RSA 問題が hard ならば, RSA-KEM は IND-CCA 安全性をみたす. ここで, $H = KDF$ とし, KDF は $\mathcal{RO}$ ランダムオラクルとする.

3.4.3 RSA-KEM の $\mathcal{TRO}$ モデルでの安全性

RSA-KEM は $\mathcal{RO}$ モデルで安全だが, $\mathcal{TRO}$ モデルでは安全ではない. 具体的には, RSA-KEM は $\mathcal{TRO}$ モデルで IND-CCA より弱い安全性である IND-CPA を満たさないことを示す. ここで, IND-CPA は攻撃者が復号クエリできない点を除いて IND-CCA と同じである.

定理 3.5 RSA 問題が hard, KDF が $\mathcal{TRO}$ の場合, RSA-KEM は IND-CPA を満たさない.

証明. まず, IND-CPA 攻撃者 $\mathcal{A}$ を以下に構成する. ここで, KDF は $\mathcal{TRO}$ である.

入力： 公開鍵 (n, e)
出力： 推定ビット b'
Step 1： $state$ を返信し, 次にチャレンジとして (K_b^*, C_0^*) を受け取る. そして, KDF の $\mathcal{TO}$ に K_b^* をクエリし, $\{r\}$ を得る.
Step 2： $\{r\}$ 内の任意の r について, $r^e \stackrel{?}{\equiv} C_0^* \pmod{n}$ をチェックする. もし, この関係を満たす r^* が存在するならば, $b' = 0$ を出力し, そうでなければ, $b' = 1$ を出力する.

次に, $\mathcal{A}$ の成功確率を評価する. チャレンジ暗号文 C_0^* が生成された場合, RSA-KEM の仕様から, $K_0^* = KDF(r^*)$ となる r^* は KDF の $\mathcal{RO}$ に質問される. よって, $\mathcal{L}_{KDF}$ は

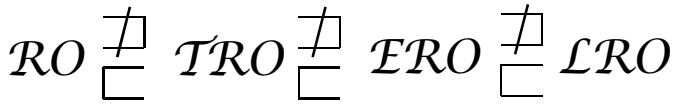

図.3.1　$\mathcal{RO}$, $\mathcal{ERO}$, $\mathcal{ERO}$, $\mathcal{RO}$ の関係性

(r^*, C_0^*, K_0^*) を含む. もし (r^*, C_0^*, K_b^*) が $\mathcal{L}_{KDF}$ に含まれないならば, $b = 1$ となる. よって, $\mathcal{A}$ は IND-CPA game に勝つ.

■

3.4.4　$\mathcal{ERO}$ モデルでの RSA-KEM の安全性

以下の通り, RSA-KEM は $\mathcal{ERO}$ モデルで安全となる.

定理 3.6 もし RSA 問題が (t', ϵ')-hard ならば, RSA-KEM は以下の通り, (t, ϵ)-IND-CCA を満たす: $t' = t + (q_{RKDF} + q_{EKDF}) \cdot expo$, $\epsilon' \geq \epsilon - \frac{q_D}{n}$. ここで, KDF は $\mathcal{ERO}$ であり, q_{RKDF} は KDF の $\mathcal{RO}$ へのクエリ回数, q_{EKDF} は KDF の $\mathcal{EO}$ へのクエリ回数, q_D は $\mathcal{DO}$ へのクエリ回数, $expo$ は n のべき上にかかるステップ数である.

この証明は 3.7 で与える

3.5　$\mathcal{LRO}$, $\mathcal{TRO}$, $\mathcal{ERO}$, $\mathcal{RO}$ の関係性

本章で, $\mathcal{LRO}$, $\mathcal{TRO}$, $\mathcal{ERO}$, $\mathcal{RO}$ の関係性を示す. 図 3.1 は以下に示す $\mathcal{LRO}$, $\mathcal{TRO}$, $\mathcal{ERO}$, $\mathcal{RO}$ の関係性を示している.

3.5.1　$\mathcal{LRO}$ vs. $\mathcal{TRO}$

$\mathcal{LRO}$ は $\mathcal{TRO}$ より多くの $\mathcal{RO}$ のリスト $\mathcal{L}_{\mathcal{RO}}$ の情報を漏らすので, $\mathcal{TRO}$ は $\mathcal{LRO}$ より強いオラクルのため, 以下の定理が成り立つ.

定理 3.7 あるシミュレータ S が存在し, 任意の識別者 $\mathcal{D}$ に対して以下が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathcal{TRO},\mathcal{LRO},S} = 0$$

ここでシミュレータの動作時間は $t_S = O(q^2)$ である.

証明. $\mathcal{LRO}$ を用いて $\mathcal{TO}$ をシミュレートする S を以下に構成する.

Simulator $S(z)$
リスト $\mathcal{L}_{\mathcal{RO}}$ を $\mathcal{LO}$ から入手する.
任意の $(x, z) \in \mathcal{L}_{\mathcal{RO}}$ に対し, $T \xleftarrow{\cup} x$.
T を返す.

上記の手続きは, $\mathcal{TO}$ の手続きそのものなので, $(\mathcal{RO}, S)$ と $(\mathcal{RO}, \mathcal{TO})$ は同じとなる. よって, 定理が成り立つ.

■

定理 3.8 $\mathcal{LRO} \not\sqsubseteq \mathcal{TRO}$.

証明. S を $\mathcal{LO}$ をシミュレートする任意のシミュレータとする. そして, $\mathcal{LRO}$ と $\mathcal{TRO}$ を識別する以下の識別者 $\mathcal{D}$ を構成する. ここで, $\mathcal{D}$ は (L, R) と対話しているとし, Real World では $(L, R) = (\mathcal{RO}, \mathcal{LO})$, Ideal World では $(L, R) = (\mathcal{RO}, S)$ である.

Distinguisher $\mathcal{D}$
01 $m \xleftarrow{\$} \{0,1\}^n$.
02 $z \leftarrow L(m)$.
03 R からリスト L を入手する.
04 もし $L \neq \{(m, z)\}$ ならば 0 を返す.
05 それ以外は 1 を返す.

ここで, m は $\{0,1\}^n$ からランダムに選ばれるため, S はこの値を知ることはできない. よって, Real World では, $\mathcal{D}$ は確率 1 で 1 を出力する. そして, Ideal World では, m と z はランダムに選ばれるため, 無視できる確率を除いて, $\mathcal{D}$ は 0 を出力する.

■

$\mathcal{LRO} \not\sqsubseteq \mathcal{TRO}$ なので, $\mathcal{TRO}$ モデルで安全で, $\mathcal{LRO}$ モデルで安全ではない暗号アルゴリズムが存在する. ここで, OAEP 暗号は $\mathcal{LRO}$ モデルでは安全ではないことが示されおり [63], 本稿で, $\mathcal{TRO}$ モデルで安全であることを示したため (3.3 章を参照), OAEP 暗号は $\mathcal{LRO}$ と $\mathcal{TRO}$ の separation となっている.

3.5.2 $\mathcal{TRO}$ vs. $\mathcal{ERO}$

$\mathcal{TRO}$ は $\mathcal{ERO}$ より多くの $\mathcal{RO}$ のリスト $\mathcal{L}_{\mathcal{RO}}$ の情報を漏らすので, $\mathcal{ERO}$ は $\mathcal{TRO}$ より強いオラクルである. 具体的には, 以下の定理と定理 3.3 が成り立つ.

定理 3.9 $\mathcal{TRO} \not\subseteq \mathcal{ERO}$.

証明. S を $\mathcal{TO}$ をシミュレートする任意のシミュレータとする. そして, $\mathcal{TRO}$ と $\mathcal{LRO}$ を識別する以下の識別者 $\mathcal{D}$ を構成する. ここで, $\mathcal{D}$ は (L, R) と対話しているとし, Real World では $(L, R) = (\mathcal{RO}, \mathcal{TO})$, Ideal World では $(L, R) = (\mathcal{RO}, S)$ である.

Distinguisher D
01 $m \xleftarrow{\$} \{0,1\}^n$.
02 $z \leftarrow L(m)$.
03 $m^* \leftarrow R(z)$.
04 If $m^* = m$, returns 1.
05 Otherwise returns 0.

m は $\{0,1\}^n$ からランダムに選ばれるため, S はこの値を知ることはできない. よって, Ideal World で, $\mathcal{D}$ は確率 1 で 1 を出力する. そして, Real World では, $\mathcal{D}$ は確率ほぼ 1 で 0 を出力する. よって, 上記の定理が成り立つ.

■

$\mathcal{TRO} \not\subseteq \mathcal{ERO}$ なので, $\mathcal{ERO}$ モデルで安全で, $\mathcal{TRO}$ モデルで安全ではない暗号アルゴリズムが存在する. ここで, RSA-KEM は $\mathcal{TRO}$ モデルでは安全ではなく (3.4 章参照), $\mathcal{TRO}$ モデルで安全であるため (3.4 章参照), RSA-KEM は $\mathcal{TRO}$ と $\mathcal{ERO}$ の separation となっている.

3.5.3 $\mathcal{ERO}$ vs. $\mathcal{RO}$

$\mathcal{ERO}$ は $\mathcal{RO}$ より多くの $\mathcal{RO}$ のリスト $\mathcal{L}_{\mathcal{RO}}$ の情報を漏らすので, $\mathcal{RO}$ は $\mathcal{ERO}$ より強いオラクルである. 具体的には, 以下の定理を示す.

定理 3.10 あるシミュレータ S, 任意の識別者 $\mathcal{D}$ に対して, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathcal{RO},\mathcal{ERO},S}(\mathcal{D}) = 0$$

ここで, シミュレータの動作時間は $t_S = O(1)$ である.

証明. シミュレータ S を何もしないように定義する. この場合, 定理が成り立つ.

■

定理 3.11 $\mathcal{ERO} \not\sqsubseteq \mathcal{RO}$.

証明. S を $\mathcal{EO}$ をシミュレートする任意のシミュレータとする. そして, $\mathcal{ERO}$ と $\mathcal{RO}$ を識別する以下の識別者 $\mathcal{D}$ を構成する. ここで, $\mathcal{D}$ は (L, R) と対話しているとし, Real World では $(L, R) = (\mathcal{RO}, \mathcal{EO})$, Ideal World では $(L, R) = (\mathcal{RO}, S)$ である.

Distinguisher D
01 $m \xleftarrow{\$} \{0,1\}^n$, $m' \xleftarrow{\$} \{0,1\}^n$.
02 $z \leftarrow L(m)$, $z' \leftarrow L(m||m')$.
03 $z^* \leftarrow R(m, z)$.
04 If $z^* = z'$, returns 1.
05 Else returns 0.

m は乱数値なので, S は m を知ることはできない. よって, Ideal World では, D は確率 1 で 1 を出力する. 一方で, Real World では, D は確率ほぼ 1 で 0 を出力する. よって, 上記の定理が成り立つ.

■

$\mathcal{ERO} \not\sqsubseteq \mathcal{RO}$ なので, $\mathcal{RO}$ モデルで安全で, $\mathcal{ERO}$ モデルで安全ではない暗号アルゴリズムが存在する. 2.5 章で Secret Prefix MAC が MD^h と $\mathcal{RO}$ の separation となることを示したが, MD^h と $\mathcal{ERO}$ は等価なので Secret Prefix MAC は $\mathcal{ERO}$ と $\mathcal{RO}$ の separation にもなっている.

3.6 定理 3.4 の証明

r^* を, 以下の通り c^* と m_b を定義するランダムテープとする.

$$r^* = H(x^*) \oplus y^* \text{ and } G(r^*) = x^* \oplus (m_b||0^{k_1}).$$

まずはじめに, OAEP の IND-CCA の環境を H の $\mathcal{TO}$ への Trace クエリに注目した返還を行う. IND-CCA の初期の環境 Exp0 では, 攻撃者 $\mathcal{A}$ は $r^* \oplus y^*$ を H の $\mathcal{TO}$ にクエリする可

能性があるが, 変換された環境 Exp1 では, $\mathcal{A}$ がそのようなクエリをした場合停止する. Succ0 と Succ1 を $\mathcal{A}$ が 1 ビット b を Exp0 と Exp1 で推定するのに成功する確率とする. 以下, これらの確率の差を評価する.

Exp0 では, $\mathcal{A}$ について次の 2 つのシナリオを考えられる.

1 つ目は, H の $\mathcal{TO}$ に x^* を得るために trace クエリ $r^* \oplus y^*$ をして, G の $\mathcal{TO}$ に推定したビット b' を用いた trace クエリ $x^* \oplus m_{b'}||0^{k_1}$ をする. もし, G の $\mathcal{TO}$ が $\perp$ を返した場合, 攻撃者は $b' \neq b$ であることを知ることができ, ゲームに勝つことができる. そして, もし, H の $\mathcal{TO}$ が複数の x^* を返した場合, $\mathcal{A}$ は G の $\mathcal{TO}$ に推定したビット b' と全ての x^* に対するトレースクエリ $x^* \oplus m_{b'}||0^{k_1}$ を行う. この時, もし G の $\mathcal{TO}$ が r^* を返した場合, 攻撃者は $b' = b$ となることを知ることができ, このゲームに勝つことができる. ここで, このシナリオでは $\mathcal{A}$ は G の $\mathcal{RO}$ にハッシュクエリ r^* を投げない, または G の $\mathcal{TO}$ に $G(r^*)$ を投げないことに注意されたい.

もう 1 つは, G の $\mathcal{RO}$ に r^* をクエリする, または, G の $\mathcal{TO}$ に $G(r^*)$ をクエリする場合である. ここで, $\mathcal{A}$ は $G(r^*)$ が $\{0,1\}^{n+k_1}$ からランダムに選ばれた値以外の情報は得られない. よって, 1 つ目のシナリオが起こる確率は, 乱数 r^* と $\mathcal{A}$ のランダムテープから, H が $\mathcal{RO}$ ということを用いると, $q_{TH} \cdot 2^{-k_0}$ でバウンドされる.

一方で, Exp1 では, H の $\mathcal{TO}$ に $r^* \oplus y^*$ をクエリした場合に停止するため, $\mathcal{A}$ は 2 つのシナリオだけを考えればよい. すなわち, G の $\mathcal{RO}$ に r^* をクエリする場合と G の $\mathcal{TO}$ に $G(r^*)$ をクエリする場合である. よって, Exp0 と Exp1 に差が出る場合, $\mathcal{A}$ が H の $\mathcal{TO}$ に $r^* \oplus y^*$ をクエリするイベントが起こる. よって, $|\mathsf{Succ1} - \mathsf{Succ0}| \leq q_{TH} \cdot 2^{-k_0}$ となる.

$\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ を Exp1 の環境で OAEP 暗号の IND-CCA を破る攻撃者とする. そして, trapdoor permutation f の (t', ϵ')-partial-domain one-wayness を破るインバータ $\mathcal{I}$ を構成する. すなわち, (f, Dom, y) が与えられた場合, $y = f(s,t)$ となる s を出力する $\mathcal{I}$ を構成する. ここで, $\mathcal{A}$ は H, G, $\mathcal{DO}$ に繰り返してクエリしないとする. $\mathcal{L}_H$ と $\mathcal{L}_G$ をそれぞれ H の $\mathcal{RO}$ と G の $\mathcal{RO}$ のハッシュリストとする. $\mathcal{L}_H$ は 3 つ組 (δ_i, h_i) $(0 \leq i \leq q_{RH})$ から構成されており, $\mathcal{L}_G$ は 3 つ組 (γ_i, g_i) $(0 \leq i \leq q_G)$ から構成されている. そして, $\mathcal{I}$ を以下に定義する.

入力 : (f, Dom, c^*) s.t. $(f, f^{-1}, Dom) \leftarrow \mathcal{G}(1^k)$ and $c^* = f(x^*, y^*)$ for $(x^*, y^*) \xleftarrow{R} \{0,1\}^{n+k_1} \times \{0,1\}^{k_0}$

出力 : x^* s.t. $(x^*, y^*) = f^{-1}(c^*)$

公開鍵を入力: f を $\mathcal{A}$ に送る.

$\mathcal{RO}$ H **simulation :** $\mathcal{A}$ が δ_i を H の $\mathcal{RO}$ にクエリした場合, 以下の振る舞いをする. もし, $(\delta_i, h_i) \in \mathcal{L}_H$ となる h_i が存在するならば, 任意の $(\gamma_j, g_j) \in \mathcal{L}_G$ に対して, $\mathcal{I}$ は $y = \gamma_j \oplus h_i$ を計算し, $c^* = f(\delta_i, y)$ が成り立つかチェックする. もし, ある γ_j でこの関係が成り立つならば, $\mathcal{I}$ は $(x^*, y^*) = f^{-1}(c^*)$ となる $\delta_i = x^*$ を得ることができる.

<If $(\delta_i, *) \notin \mathcal{L}_H$ >
$h_i \in \{0,1\}^{k_0}$ を生成する. そして, (δ_i, h_i) を $\mathcal{L}_H$ に追加して, h_i を $\mathcal{A}$ に返信する.
<If $(\delta_i, *) \in \mathcal{L}_H$ >
$\mathcal{L}_H$ から δ_i に対応した h' を見つけ, h' を $\mathcal{A}$ に返信する.

$\mathcal{RO}$ G **simulation :** $\mathcal{A}$ が G の $\mathcal{RO}$ に γ_i をクエリした場合, 以下の動作をする:
任意の $(\delta_j, h_j) \in \mathcal{L}_H$ に対して, $\mathcal{I}$ は $y = \gamma_i \oplus h_j$ を計算し, $c^* = f(\delta_j, y)$ が成り立つかいかなをチェックする. もしある γ_j がこの関係を満たすならば, $\mathcal{I}$ は $(x^*, y^*) = f^{-1}(c^*)$ となる $\delta_j = x^*$ を得ることができる.
<If $(\gamma_i, *) \notin \mathcal{L}_G$ >
$g_i \in \{0,1\}^{n+k_1}$ を生成する. (γ_i, g_i) を $\mathcal{L}_G$ に追加し, g_i を $\mathcal{A}$ に返信する.
<If $(\gamma_i, *) \in \mathcal{L}_G$ >
$\mathcal{L}_G$ から γ_i に対応する g' を見つけ, g' を $\mathcal{A}$ に返信する.

$\mathcal{DO}$ **simulation :** $\mathcal{A}$ が $\mathcal{DO}$ に復号クエリ c_i を投げ的亜場合, 次の動作をする:
$\mathcal{L}_H$ から (δ, h) を見つけ, $\mathcal{L}_G$ から $c_i = f(\delta, \gamma \oplus h)$, $[g \oplus \delta]_{k_1} = 0^{k_1}$ となる (γ, g) を見つける. もし, この条件をみたすペアが存在する場合, $[g \oplus \delta]^n$ を返信する. そうでなければ, $reject$ を返信する.

$\mathcal{TO}$ H **simulation :** $\mathcal{A}$ が H の $\mathcal{TO}$ に h' をクエリした場合, $\mathcal{L}_H$ から h' を探す. もし $\{(\delta, h')\}$ が存在する場合, $\mathcal{A}$ に全ての入力値 $\{\delta\}$ を返す. そうでなければ, $\perp$ を返す.

$\mathcal{TO}$ G **simulation :** $\mathcal{A}$ が G の $\mathcal{TO}$ に g' をクエリした場合, $\mathcal{L}_G$ から g' を探す. もし $\{(\gamma, g')\}$ が存在する場合, $\mathcal{A}$ に全ての入力値 $\{\gamma\}$ を返す. そうでなければ, $\perp$ を返す.

Challenge ciphertext : $\mathcal{A}$ が (m_0, m_1) を出力した場合, 1 ビットの値 b をランダムに選んで, c^* を m_b の暗号文として返す.

Finalization : $\mathcal{A}$ が b' を出力した場合, もし $\mathcal{I}$ が $\mathcal{RO}$ のシミュレーションで x^* を得ていた場合, x^* を返す. そうでなければ, $\mathcal{L}_H$ から δ を選んで, δ を出力する.

次に, $\mathcal{I}$ の成功確率を評価する. Finalization で, もし x^* が $\mathcal{A}$ によって H の $\mathcal{RO}$ にクエリされていた場合, $\mathcal{I}$ は x^* を得ることができる. しかし, $\mathcal{A}$ は以下のケースが起きた場合, Real 環境とシミュレーション環境を識別することができる.

1. チャレンジ暗号文を c^* に変更することで, $\mathcal{A}$ はこのチャレンジ暗号文が正しくないものだと気づく.
2. シミュレーション環境では, $\mathcal{I}$ は復号クエリに対して $reject$ を返すが, Real 環境では, 質問された暗号文が正しいと知られていて, $\mathcal{DO}$ がこの平文を返す.

前者のケースの場合, $\mathcal{A}$ は G の $\mathcal{RO}$ に r^* を質問する, または, G の $\mathcal{TO}$ に $x^* \oplus (m_b||0^{k_1})$ を質問しなければならない. この理由は, $\mathcal{A}$ はこれらのクエリのいずれも質問しなければ, $G(r^*)$ は $\{0,1\}^{n+k_1}$ の乱数値, m_0, m_1 と独立な値となるからである.

後者のケースの場合, r が G の $\mathcal{RO}$ にクエリされないまたは x が H の $\mathcal{RO}$ にクエリされないのどちらかが起こらず, そして, c が正しい暗号文の場合のみ, $c = f(x, y)$, $x = (m||0^{k_1}) \oplus G(r)$, $y = r \oplus H(x)$ となり, $\mathcal{I}$ は暗号文 c に関するシミュレーションに失敗する.

そして, $\mathcal{I}$ の成功確率を評価するために以下のイベントを考える.

- AskH : H の $\mathcal{RO}$ に x^* がクエリされていた場合.
- AskRG : G の $\mathcal{RO}$ に $x^* \oplus (m_b||0^{k_1})$ がクエリされていた場合.
- AskTG : G の $\mathcal{TO}$ に $x^* \oplus (m_b||0^{k_1})$ がクエリされていた場合.
- RGBad : r^* が G の $\mathcal{RO}$ にすでにクエリされていて, その答えが $x^* \oplus (m_b||0^{k_1})$ と異なってた場合.
- TGBad : $x^* \oplus (m_b||0^{k_1})$ が G の $\mathcal{RO}$ にすでにクエリされていて, その答えが r^* と異なってた場合.
- DBad : $\mathcal{I}$ が $\mathcal{DO}$ のシミュレーションに失敗した場合.
- Bad = RGBad $\vee$ TGBad $\vee$ DBad

ここで, RGBad ならば AskRG, TGBad ならば AskTG となることに注意されたい.

$\mathcal{I}$ の成功確率は AskH より, $\epsilon' \geq \Pr[\mathsf{AskH}] \cdot {q_{RH}}^{-1}$ となる. よって, イベント Bad を用いて AskH を場合わけすることで以下の式で $\Pr[\mathsf{AskH}]$ を評価する.

$$\Pr[\mathsf{AskH}] = \Pr[\mathsf{AskH} \wedge \mathsf{Bad}] + \Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}].$$

まず, 1 番目の式を評価する. ここで, $\mathsf{Bad} = \mathsf{RGBad} \vee \mathsf{TGBad} \vee \mathsf{DBad}$, $\Pr[\mathsf{RGBad}] \leq \Pr[\mathsf{AskRG}]$, $\Pr[\mathsf{TGBad}] \leq \Pr[\mathsf{AskTG}]$ となることに注意されたい.

$$\begin{aligned}
\Pr[\mathsf{AskH} \wedge \mathsf{Bad}] &= \Pr[\mathsf{Bad}] - \Pr[\mathsf{Bad} \wedge \neg\mathsf{AskH}] \\
&\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{RGBad} \wedge \neg\mathsf{AskH}] \\
&\quad - \Pr[\mathsf{TGBad} \wedge \neg\mathsf{AskH}] \\
&\quad - \Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}] \\
&\geq \Pr[\mathsf{Bad}] - \Pr[\mathsf{AskRG} \wedge \neg\mathsf{AskH}] \\
&\quad - \Pr[\mathsf{AskTG} \wedge \neg\mathsf{AskH}] \\
&\quad - \Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}].
\end{aligned}$$

ここで, Exp1 の $\mathcal{A}$ は $\neg$AskH によって $r^* = H(x^*) \oplus y^*$ を知ることができないため $\Pr[\mathsf{AskRG} \wedge \neg\mathsf{AskH}] \leq q_{RG} \cdot 2^{-k_0}$ となる. そして, 同様の理由から $\Pr[\mathsf{AskTG} \wedge \neg\mathsf{AskH}] \leq q_{TG} \cdot 2^{-(n+k_1)}$ となる. また, $\mathcal{TO}$ はハッシュリストをアップデートしないため, $\Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}]$ の評価は [30] 内の評価と同様の評価を適用できる. よって, $\Pr[\mathsf{DBad} \wedge \neg\mathsf{AskH}] \leq q_D(2 \cdot 2^{-k_1} + (2q_{RG}+1) \cdot 2^{-k_0})$ となる. 以上より, $\Pr[\mathsf{AskH} \wedge \mathsf{Bad}] \geq \Pr[\mathsf{Bad}] - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}}$ となる.

次に, 2 番目の確率を評価する. この評価は, Bad でイベント分けをせずに評価できるため, [30] の評価を適用できる. よって, [30] の評価から $\Pr[\mathsf{AskH} \wedge \neg\mathsf{Bad}] \geq \left(\frac{\epsilon}{2} + \frac{1}{2} - \Pr[\mathsf{Bad}]\right) - \frac{\Pr[\neg\mathsf{Bad}]}{2} = \frac{\epsilon - \Pr[\mathsf{Bad}]}{2}$ となる.

以上の解析と $\Pr[\mathsf{Bad}] \geq 0$ より, $\Pr[\mathsf{AskH}] \geq \frac{\epsilon}{2} - \frac{2q_D q_{RG} + q_D + q_{RG}}{2^{k_0}} - \frac{2q_D}{2^{k_1}} - \frac{q_{TG}}{2^{n+k_1}}$ となる.

また, $\mathcal{I}$ の動作時間は [30] で評価されたものを適用でき, $t' = t + q_{RG} \cdot q_{RH} \cdot perm$ となる.

3.7 定理 3.6 の証明

まず, RSA-KEM の IND-CCA 環境を $\mathcal{DO}$ が攻撃者にアドバンテージを与えないように変更する.

Exp0 を初期の環境として, Succ0 をこの環境で攻撃者 $\mathcal{A}$ が b の推定に成功する確率とする. この場合, $\mathcal{A}$ は $c_0^* = {r^*}^e$ を満たすチャレンジ (K_b^*, c_0^*) を受け取る.

次に, Exp1 を, $\mathcal{A}$ が c_0^* をチャレンジ暗号文を受け取る前に $\mathcal{DO}$ にクエリしたとき以外は Exp0 と同じになる環境とする. ここで, Exp1 ではこの場合, 停止する. Succ1 を, Exp1 で $\mathcal{A}$ が b の推定に成功する確率とする. この変換は文献 [52] の $\mathcal{RO}$ モデルでの証明でも同様の変換が出てきて, この解析を適用すると, $|\mathsf{Succ1} - \mathsf{Succ0}| \leq q_D/n$ となる.

Exp2 を, (K_b^*, c_0^*) を初めに生成し, $\mathcal{A}$ が r^* を $\mathcal{RO}$ にクエリしたときに停止して, それ以外は, Exp1 と同じとする. そして, Succ2 を $\mathcal{A}$ が Exp2 で b の推定に成功する確率とし, E をこのケースで環境が停止するイベントとする. この場合, 文献 [52] の証明と同様に, $\Pr[E]$ の評価を RSA 問題が破られる確率で抑えることができる. これは以下で示す.

補題 9 もし, 時間 t'' で動く $\mathcal{A}$ が確率 ϵ'' でイベント E を起こせるならば, 時間 t' で動き, 確率 ϵ' で RSA 問題を解くことができるインバータ $\mathcal{I}$ を構成できる. ここで,

$$t' = t'' + (q_{RH} + q_{EH}) \cdot expo,$$
$$\epsilon' = \epsilon''.$$

証明. $\mathcal{A}$ は $\mathcal{ERO}$ と $\mathcal{DO}$ に同じクエリをしないと仮定する. H を H の $\mathcal{RO}$ の $\mathcal{L}_H$ ハッシュリストとしする. $\mathcal{L}_H$ は 3 つ組 (δ_i, y_i, h_i) $(0 \leq i \leq q_{RH} + q_D + q_{EH})$ から構成されるとする. ここで, y_i はこの証明のシミュレーションで用いる中間値である. $\mathcal{L}_{\mathcal{EO}}$ を H の $\mathcal{EO}$ のリストとする. $\mathcal{L}_{\mathcal{EO}}$ は 3 つ組 (δ_i, h_i, z_i) $(0 \leq i \leq q_{EH})$ から構成される. そして, 具体的な $\mathcal{I}$ の構成は以下の通りである.

入力 : (n, e, y^*). ここで, n を RSA の法とする. e をべき指数とし, $\gcd(e, \phi(n)) = 1$, $y^* \xleftarrow{R} \mathbb{Z}_n$ とする.

出力 : x^* s.t. $x^* \equiv {y^*}^d \pmod{n}$

公開鍵の入力 : Exp2 内の公開鍵を入力する操作として, $\mathcal{A}$ に (n, e) を送る.

$\mathcal{DO}$ simulation : $\mathcal{A}$ が y_i を $\mathcal{DO}$ にクエリする場合, $\mathcal{I}$ は以下の動作を行う:
$\mathcal{I}$ は $y_i \in \mathbb{Z}_n$ が成り立つかチェックする. もし成り立たなければ, $\mathcal{I}$ はエラーシンボル $\perp$ を返す. そうでなければ, $\mathcal{I}$ は $y_i = \delta_i^e$ となる (δ_i, y_i, h_i) を $\mathcal{L}_H$ から探す. もしこの 3 つ組 (δ_i, y_i, h_i) が見つかった場合, $\mathcal{I}$ は h_i を返す. 見つからなければ, $\mathcal{I}$ は $h_i \in \{0,1\}^k$ を生成し, $(\emptyset, y_i, h_i)$ を $\mathcal{L}_H$ に加えて, h_i を返信する.

$\mathcal{RO}$ simulation : $\mathcal{A}$ が δ_i を $\mathcal{RO}$ にクエリした場合, 以下の動作を行う:

<If $y_i = y^*$ s.t. $y_i = \delta_i^e \mod n$ >
$\mathcal{I}$ は停止し, RSA 問題の出力のステップに飛ぶ.
<If $(\delta_i, *, h_i) \in \mathcal{L}_H$ >
h_i を返す.
<If $(\delta_i, *, *) \notin \mathcal{L}_H$ and $(\emptyset, y_i, h_i) \in \mathcal{L}_H$ s.t. $y_i = \delta_i^e \mod n$ >
$\mathcal{L}_H$ 内の $(\emptyset, y_i, h_i)$ を (δ_i, y_i, h_i) に変更して, h_i を返す.
<If $(\delta_i, *, *) \notin \mathcal{L}_H$ and $(\emptyset, y, h) \notin \mathcal{L}_H$ s.t. $y = \delta_i^e \mod n$ >
$y_i = \delta_i^e \mod n$ を計算し, $h_i \in \{0,1\}^k$ を生成し, (δ_i, y_i, h_i) を追加し, h_i を返信する.

$\mathcal{EO}$ simulation : $\mathcal{A}$ が (δ_i, h_i) を $\mathcal{EO}$ にクエリした場合, 以下の動作を行う:
$(\delta_i, h_i, *)$ を $\mathcal{L}_{\mathcal{EO}}$ から探索する. もし, $h_i = K_b^*$ ならば, 乱数 $z_i \in \{0,1\}^k$ を選んで, (δ_i, K_b^*, z_i) を追加する. もし, この条件を満たす (δ_i, h_i, z_i) が存在したならば, z_i を返す. もし, $(\delta', *, h_i)$ となる 3 つ組が $\mathcal{L}_H$ に 1 つだけ存在したならば, $z_i \in \{0,1\}^k$ を生成 s, $y_i = (\delta'||\delta_i)^e \mod n$ を計算し, $(\delta'||\delta_i, y_i, z_i)$ を $\mathcal{L}_H$ に追加し, (δ_i, h_i, z_i) を $\mathcal{L}_{\mathcal{EO}}$ に追加し, z_i を返す. それ以外は, $z_i \in \{0,1\}^k$ を生成し, (δ_i, h_i, z_i) を $\mathcal{L}_{\mathcal{EO}}$ に追加し, z_i を返す.

Challenge ciphertext : $\mathcal{A}$ が $(state)$ を出力した場合, 暗号化の手続きで (K^*, y') を計算して, (K^*, y') をチャレンジとして返す.

Output : $\mathcal{RO}$ simulation の <if $y_i = y^*$ s.t. $y_i = \delta_i^e \mod n$ > のケースで $\mathcal{I}$ は x^* として δ_i を出力する.

次に, $\mathcal{I}$ の成功確率を評価する. $\mathcal{RO}$ simulation では, もし, $y_i = \delta_i^e \mod n$ が成り立ち $y_i = y^*$ となるならば, $\mathcal{I}$RSA 問題を解くことに成功する. このイベントは Exp2 の E と同じである. $\mathcal{EO}$ simulation では, もし $\mathcal{A}$ がある x について (x, K_b^*) をクエリした場合, $\mathcal{I}$ は, $b = 0$ の場合でも, $H(r^*||x)$ を返す代わりに乱数を返す. しかし, このシミュレーションは以下の補助命題より $\mathcal{EO}$ の Real interface と識別ができない. ここで, 実際の $\mathcal{EO}$ とシミュレーションを識別しようとする識別者 $\mathcal{D}$ を考える.

補題 10 もし, H の $\mathcal{RO}$ の出力が入力と独立に選ばれるならば, $\mathcal{D}$ は $\mathcal{EO}$ とこのシミュレーションを識別できない.

証明. もし, $\mathcal{EO}$ とシミュレーションを識別できる $\mathcal{D}$ が存在するならば, H の $\mathcal{RO}$ の出力と乱数値を見分けることができるアルゴリズム $\mathcal{ALG}$ が存在することを示す.

> **Step 1 :** $\mathcal{D}$ がある x について, (x, K_b^*) を $\mathcal{EO}$ にクエリするケースを除いて, Exp2 と同じ環境を $\mathcal{D}$ に対してシミュレートする.
>
> **Step 2 :** ある x について $\mathcal{EO}$ に対するクエリ (x, K_b^*) を受け取った場合, $r^*||x$ を H の $\mathcal{RO}$ にクエリし, z を $H(r^*||x)$ または乱数 $rand$ として受け取り, z を $\mathcal{D}$ に返す.
>
> **Step 3 :** もし, $\mathcal{D}$ が $\mathcal{EO}$ と対話していると決めたならば, z を $H(r^*||x)$ として, そうでなければ, z を $rand$ とする.

$z = H(r^*||x)$ の場合, $\mathcal{D}$ のインターフェースは $\mathcal{EO}$ と同じである. また, $z = rand$ の場合 $\mathcal{D}$ のインターフェースはシミュレーションのインターフェースと同じである. よって, もし $\mathcal{D}$ が識別に成功すれば, $\mathcal{ALG}$ も同様に成功する.

■

以上より, $\mathcal{I}$ は $\mathcal{A}$ に対して Exp2 の環境を完全にシミュレートできている. よって,

$$\epsilon' = \epsilon''.$$

そして, $\mathcal{I}$ は n のべき乗を最大 $q_{RH} + q_{EH}$ 回計算することから, 次の式が成り立つ.
$t' = t'' + (q_{RH} + q_{EH}) \cdot expo.$

■

Exp1 と Exp2 は E が起こるまで同じである. よって, $|\mathsf{Succ2} - \mathsf{Succ1}| = \epsilon'$ となる.

鍵 K_b^* は $\mathcal{A}$ が Exp2 で得ることができる情報とは独立であることから, $\mathcal{A}$ は b の情報を得ることができない. よって, $\mathsf{Succ2} = 1/2$. また, $\mathsf{Succ0} \leq |\mathsf{Succ1} - \mathsf{Succ0}| + |\mathsf{Succ2} - \mathsf{Succ1}| + \mathsf{Succ2}$ なので, $\mathsf{Succ0} \leq \epsilon' + \frac{q_D}{n} + 1/2$ が成り立つ. よって, $\epsilon' \geq \epsilon - \frac{q_D}{n}$ となる.

■

第 4 章

マルチステージゲームで IFRO 安全なハッシュ関数の救済

4.1 はじめに

4.1.1 Indifferentiability

Maurer, Renner, Holenstein (MRH) が提案した Indifferentiability の定理 (または, MRH 定理と呼ばれている) は全てのシングルステージゲームで定義される安全性 $\mathbf{G}_s$ とその安全性をもとに設計された全ての暗号アルゴリズム $\mathbf{C}$ をカバーする. 具体的には, $\forall \mathcal{G} \in \mathbf{G}_s, \forall \mathcal{C} \in \mathbf{C}$: 「$F_1$ is indifferentiable from F_2 ($F_1 \sqsubset F_2$ と書く)」ならば, 「$\mathcal{C}(F_1)$ は $\mathcal{C}(F_2)$ の $\mathcal{G}$-安全と同じ安全性を持つ ($\mathcal{C}(F_1) \succ_{\mathcal{G}} \mathcal{C}(F_2)$ と書く)」.

MRH 定理.[40] $F_1 \sqsubset F_2 \Rightarrow \forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{G} \in \mathbf{G}_s$: $\mathcal{C}(F_1) \succ_{\mathcal{G}} \mathcal{C}(F_2)$.

ここで, $F_1 \sqsubset F_2$ の安全性ゲームは, 攻撃者がアクセスできるインターフェース $F.adv$ とそれ以外の正直なパーティーがアクセスできるインターフェース $F.hon$ を考えて, あるシミュレータ S に対して, $(F_1.hon, F_1.adv)$ と $(F_2.hon, S^{F_2.adv})$ の識別ゲームを考える. そして, 任意の識別者 $\mathcal{D}$ が Real World $(F_1.hon, F_1.adv)$ と Ideal World $(F_2.hon, S^{F_2.adv})$ を識別できないことが示せれば $F_1 \sqsubset F_2$ となる. よって, $F_1 \sqsubset F_2$ は, S を介して, $(F_1.hon, F_1.adv)$ から得られる情報は全て $(F_2.hon, F_2.adv)$ から得られることを保証する. すなわち, S を F_2 モデルのある攻撃者 $\mathcal{A}_2$ の動作とすると, 任意の F_1 モデルの攻撃者 $\mathcal{A}_1$ が得られる情報は S を介して F_2 モデルのある攻撃者 $\mathcal{A}_2$ も得られることになる. よって, 任意の攻撃者 $\mathcal{A}_2$ に対して $\mathcal{C}(F_2)$ が $\mathcal{G}$-安全ならば, 任意の攻撃者 $\mathcal{A}_1$ に対して $\mathcal{C}(F_1)$ も同様に安全となり, $F_1 \sqsubset F_2$ が保証さ

れる.

4.1.2 $\mathcal{RO}$ 証明法

ランダムオラクル ($\mathcal{RO}$) 証明法は暗号アルゴリズム $\mathcal{C}$ にハッシュ関数 $H^{\mathcal{U}}$ を組み込んだ暗号システム $\mathcal{C}^{H^{\mathcal{U}}}$ の安全性を証明する方法で, $H^{\mathcal{U}}$ を理想化した $\mathcal{RO}$ を用いた暗号システム $\mathcal{C}^{\mathcal{RO}}$ の安全性を証明する [7]. この証明法の利点は, ハッシュ関数を理想化することで暗号システムの安全性証明が容易になる点である. そして, これまで OAEP 暗号や PSS 署名など多くの暗号アルゴリズムに対して, この証明法が適用されている.

しかし, 厳密には $\mathcal{C}(\mathcal{RO})$ の安全性は $\mathcal{RO}$ の代わりに $H^{\mathcal{U}}$ を組み込んだ $\mathcal{C}(H^{\mathcal{U}})$ の安全性を保証せず, $\mathcal{RO}$ と $H^{\mathcal{U}}$ にギャップがあることが知られている. すなわち, $\mathcal{C}(\mathcal{RO})$ は $\mathcal{G}$-安全だが, $\mathcal{C}(H^{\mathcal{U}})$ は $\mathcal{G}$-安全でない安全性 $\mathcal{G}$ と暗号アルゴリズム $\mathcal{C}$ が存在することが知られている.

そこで, Coron, Dodis, Malinaud, Puniya (CDMP) は MRH 定理を用いてこのギャップを埋めることができるハッシュ関数の設計に取り組んだ [24]. MRH 定理から $H^{\mathcal{U}} \sqsubset \mathcal{RO}$ となる $H^{\mathcal{U}}$ を用いることで, $\forall \mathcal{C} \in \mathbf{C}, \forall \mathcal{G} \in \mathbf{G}_s$: $\mathcal{C}(H^{\mathcal{U}}) \succ_{\mathcal{G}} \mathcal{C}(\mathcal{RO})$ が保証されるため, CDMP は定義域拡張構造 H として ChopMD 構造や Prefix-Free MD 構造などに理想的なプリミティブ $\mathcal{U}$ を組み込んだ時に $H^{\mathcal{U}} \sqsubset \mathcal{RO}$ となることを証明した. 以降, $H^{\mathcal{U}} \sqsubset \mathcal{RO}$ を考えた安全性を IFRO 安全性と呼ぶことにする.

CDMP が IFRO 安全な構造を提案して以降, 多くの実用的な構造が提案されている. 例えば, Sponge 構造は IFRO 安全な構造で, 次世代標準ハッシュ関数である SHA-3 の構造に採用されている [45, 13]. また, ChopMD 構造は FIPS 180-4 で標準化されている, SHA-512/224 や SHA-512/256 の構造に採用されている [46]. このように, 標準化されている多くのハッシュ関数が IFRO 安全性を考えて設計されているため, IFRO 安全性は現在のハッシュ関数の定義域拡張構造の設計で標準的な安全性となっている.

4.1.3 IFRO 安全性の否定的な結果

Ristenpart, Shacham, Shrimpton (RSS) は, IFRO 安全性はシングルステージゲームの安全性 $\mathbf{G}_s$ を保証するものの, マルチステージゲームの安全性 $\mathbf{G}_m$ を保証しないことを示した. すなわち, ある IFRO 安全なハッシュ関数 $H^{\mathcal{U}}$ に対して, 以下のことが成り立つ.

$$\exists \mathcal{G} \in \mathbf{G}_m, \exists \mathcal{C} \in \mathbf{C} \text{ s.t. } \mathcal{C}^{\mathcal{RO}} \text{ は } \mathcal{G}\text{-安全だが } \mathcal{C}^{H^{\mathcal{U}}} \text{ は } \mathcal{G}\text{-安全ではない.}$$

具体的に RSS は, $\mathcal{G}$ として 2 者間のチャレンジレスポンスプロトコルの安全性 CRP を定義して, $\mathcal{C}$ としてチャレンジレスポンスプロトコル $\mathcal{CR}$ を構成して, H として Sponge や ChopMD などに対して上記のことが成り立つことを示した.

この否定的な結果は Indifferentiability の S のステイトの条件と $\mathbf{G}_m$ で考える攻撃者のステイトの条件の違いから生じる. Indifferentiability の S はステイトフルなシミュレータであるのに対し, マルチステージゲームのそれぞれのステージの攻撃者間で引き継げるステイトに制限があるため, S の手続きはマルチステージゲームの攻撃者の動作としてみなすことができないため, Indifferentiability は $\mathbf{G}_m$ を保証せず, この理論を用いた IFRO 安全性も $\mathbf{G}_m$ を保証しない.

一方で, 上記の反例は任意のマルチステージゲームの安全性と任意の暗号アルゴリズムに対して成り立つとは限らないことに注意されたい. すなわち, あるマルチステージゲームの安全性 $\mathcal{G}$ と $\mathcal{RO}$ を用いた時に $\mathcal{G}$-安全なある暗号アルゴリズム $\mathcal{C}$ に対して, IFRO 安全な $H^{\mathcal{U}}$ を組み込んだ $\mathcal{C}^{H^{\mathcal{U}}}$ が安全性となる可能性がある. 特に, 重要なマルチステージゲームの安全性 $\mathcal{G}_m$ と $\mathcal{RO}$ を用いた時に $\mathcal{G}_m$ 安全な実用的な暗号アルゴリズム $\mathcal{C}$ に対して, 重要な構造 H を用いたハッシュ関数 $H^{\mathcal{U}}$ を $\mathcal{C}$ に組み込んだ $\mathcal{C}(H^{\mathcal{U}})$ が $\mathcal{G}_m$-安全性を満たすか否かを示すことは特に重要な研究課題と言える.

4.1.4 Reset Indifferentiability

RSS は Indifferentiability を拡張た Reset Indifferentiability を提案して, Reset Indifferentiability がマルチステージゲームの安全性をカバーすることを証明した.

> **RSS 定理. [51]** F_1 is resest indifferentiable from F_2 ($F_1 \sqsubset_r F_2$ と書く)
> $\Rightarrow \forall \mathcal{G} \in \mathbf{G}, \forall \mathcal{C} \in \mathbf{C}$: $\mathcal{C}(F_1) \succ_{\mathcal{G}} \mathcal{C}(F_2)$.

$F_1 \sqsubset_r F_2$ のゲームは, S としてステイトレスなシミュレータを考た, $(F_1.hon, F_1.adv)$ と $(F_1.hon, S^{F_1.adv})$ の識別ゲームである. ここで, $F_1 \sqsubset_r F_2$ を, 任意の識別者があるステイトレスなシミュレータ S に対して, Real World $(F_1.hon, F_1.adv)$ と Ideal World $(F_1.hon, S^{F_1.adv})$ を識別できないことと定義する. S はステイトレスなシミュレータなので, 任意の攻撃者のステイトの条件をカバーできて, MRH 定理と同様の議論により上記の RSS 定理を保証することができる.

RSS 定理の系として, $H^{\mathcal{U}} \sqsubset_r \mathcal{RO} \Rightarrow \forall \mathcal{G} \in \mathbf{G}, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{H^{\mathcal{U}}} \succ_{\mathcal{G}} \mathcal{C}^{\mathcal{RO}}$ が成り立つため, 重要な構造 H に対して, $H^{\mathcal{U}} \sqsubset_r \mathcal{RO}$ が示せれば我々が取り組む研究課題が解決できるものの, 多くの実用的な構造, 具体的には one-pass ハッシュ関数の構造 H に対して $H^{\mathcal{U}} \not\sqsubset_r \mathcal{RO}$ とな

ることが示されている [51]. One-pass ハッシュ関数の構造として, Sponge や ChopMD など実用的な構造は全て含まれるため, このアプローチに変わる他の証明法の提案が必要となる.

4.1.5 本研究成果 – $\mathcal{WRO}$ 証明法の提案 –

本研究では, $\mathcal{C}^{H^{\mathcal{U}}}$ の $\mathcal{G}_m$ 安全性を保障するために, Reset Indifferentiability を用いた「Weakened Random Oracle ($\mathcal{WRO}$) 証明法」を提案する. そして, 本稿では $\mathcal{G}_m$ として重要なマルチステージの安全性である CDA 安全性 [3, 4], $\mathcal{C}$ として実用的な暗号アルゴリズムである REwH や EwH [3, 4], H として重要な構造で Sponge 構造と ChopMD 構造を扱い, $\mathcal{WRO}$ 証明法を用いて, $\mathcal{C}(H^{\mathcal{U}})$ が $\mathcal{G}_m$ 安全となることを証明する.

$\mathcal{WRO}$ 証明法では $\mathcal{G}_m \in \mathbf{G}_m$, $\mathcal{C} \in \mathbf{C}$, $H^{\mathcal{U}}$ に対して以下のことを証明して, $\mathcal{C}^{H^{\mathcal{U}}}$ の $\mathcal{G}_m$ 安全性を保障する.

1. $\mathcal{RO}$ を弱めた $\mathcal{WRO}$ を定義して, $H^{\mathcal{U}} \sqsubset_r \mathcal{WRO}$ を証明する.
2. $\mathcal{C}^{\mathcal{WRO}}$ が $\mathcal{G}$ 安全となることを証明する.

そして, RSS 定理から上記のことが証明できると, $\mathcal{C}^{H^{\mathcal{U}}}$ の $\mathcal{G}_m$ 安全性を保証することができる.

4.1.5.1 $\mathcal{WRO}$ の定義方法.

ここで, ChopMD 構造を例に $\mathcal{WRO}$ を定義する. この例では, ChopMD 構造に組み込むプリミティブとしてランダムオラクル圧縮関数 $h : \{0,1\}^{3r} \to \{0,1\}^{2r}$ を考えて, 入力長を $2r$ bit, 出力長を r bit とする $\mathsf{ChopMD}^h(M_1 \| M_2) = chop_n(h(h(IV \| M_1) \| M_2))$ を考える. $chop_n$ は $2n$ bit の入力のうち下位 n bit を出力する関数である. ここで, 識別者 $\mathcal{D}$ は 2 つのオラクル (L, R) と対話して, Real World は $(L, R) = (\mathsf{ChopMD}^h, h)$ で, Ideal World では $(L, R) = (\mathcal{WRO}.hon, S^{\mathcal{WRO}.adv})$ である. $\mathcal{WRO} = \mathcal{RO}$, すなわち $\mathcal{WRO}.hon = \mathcal{RO}, \mathcal{WRO}.adv = \mathcal{RO}$ の場合, $\mathsf{ChopMD}^h \not\sqsubset_r \mathcal{RO}$ となることが示されており, $\mathsf{ChopMD}^h \sqsubset_r \mathcal{RO}$ となるステイトレスなシミュレータ S が構成できないため, $\mathcal{WRO}.adv$ として $\mathcal{RO}$ の情報を漏らすようなオラクル $\mathcal{O}^*$ を追加して, $\mathsf{ChopMD}^h \sqsubset_r \mathcal{WRO}$ となる S が構成できるようにする.

以下, この例を用いて $\mathcal{O}^*$ の定義方法を説明する. まず, $\mathsf{ChopMD}^h \sqsubset_r \mathcal{WRO}$ が成り立つための S が実現すべき 2 つのケースを説明して, この 2 つのケースをもとに $\mathcal{O}^*$ を定義する.

実現すべきケース 1. まず, Real World では R はランダムオラクル圧縮関数 h で, $\mathcal{D}$ が図.4.1 のように同じクエリを h に質問した場合, そのレスポンスは同じ値 $y_1 = y_2$ となる. よって, 実現すべき 1 つ目のケースとして, Ideal World でも図.4.1 の手続きで $y_1 = y_2$ となるように

識別者の手続き (ケース 1)
1. $\mathcal{D}$ が R に x をクエリして, レスポンス y_1 を受け取る.
2. $\mathcal{D}$ が R に x をクエリして, レスポンス y_2 を受け取る.

図.4.1　識別者の手続き 1

$\mathcal{D}$ の手続き (ケース 2)
1. $\mathcal{D}$ が R に (IV, m_1) をクエリして, レスポンス y_1 を受け取る.
2. $\mathcal{D}$ が R に (y_1, m_2) をクエリして, レスポンス y_2 を受け取る.
3. $\mathcal{D}$ が L に $m_1 \| m_2$ をクエリして, レスポンス z を受け取る.

図.4.2　識別者の手続き 2

することである.

このケースに対して, S への初出のクエリ x に対して, レスポンス y を定義して, クエリレスポンスペア (x, y) を S 自身が持っておけば, 同じクエリ x に対して, 同じ値 y を返せるが, S はステイトレスなので (x, y) を記録できない問題がある. そこで, $\mathcal{O}^*$ にランダムオラクル $\mathcal{RO}^*$ を加えて, $\mathcal{RO}^*$ を用いてこのケースを実現する. 具体的には, S へのクエリ x に対してレスポンス y を $y := \mathcal{RO}^*(x)$ とする. これにより, クエリ x に対するレスポンスは $\mathcal{RO}^*(x)$ の出力値なので常に同じ値となり, 1 つめのケースを実現できる.

実現すべきケース 2. 次に, 2 つめのケースについて, Real World では $(L, R) = (\mathsf{ChopMD}^h, h)$ で, ChopMD は h を用いて出力を計算しているため, L の入出力値と R の入出力値に ChopMD の構造に依存した関係がある. 具体的な例として, $\mathcal{D}$ が図.4.2 の動作を行った場合, $z = chop_n(y_2)$ となる. そして, Ideal World で以下のように S を定義するとこのケースを実現できる.

- ステップ 1 に対する手続: クエリ $IV \| m_1$ に対して S はレスポンス y_1 を定義して, y_1 を返信するとともに, $(IV \| m_1, y_1)$ を記録しておく.
- ステップ 2 に対する手続: クエリ $y_1 \| m_2$ に対して, y_1 を手掛かりに $(IV \| m_1, y_1)$ を見つけて, レスポンス y_2 を $chop_n(y_2) = \mathcal{RO}(m_1 \| m_2)$ となるように定義する.

Ideal World $(L, R) = (\mathcal{RO}, S)$ で $\mathcal{D}$ が図.4.2 の動作に対して S が上記の手続きを実行できた場合, $z = chop_n(y_2)$ となるが, ステップ 1 で $(IV \| m_1, y_1)$ を記録して, ステップ 2 で y_1 から m_1 を見つける必要があり, S はステイトレスなシミュレータなので, $(IV \| m_1, y_1)$ を記

録できないため, この手続きを実行できない. そこで, y_1 から m_1 を逆引きできるオラクルを $\mathcal{O}^*$ に追加する. 具体的には, ランダムオラクル $\mathcal{RO}^\dagger$ と Trace オラクル $\mathcal{TO}$ から構成される Traceable Random Oracle $\mathcal{TRO}$ を追加する. $\mathcal{TO}$ はクエリ y に対して, 過去に $y = \mathcal{RO}(m)$ となる m が $\mathcal{RO}$ にクエリされていた場合 m を返し, それ以外は $\perp$ を返すオラクルである. そして, $\mathcal{TRO}$ を用いて S の手続きを以下のように定義すると, ケース 2 を実現するステイトレスなシミュレータが構成できる.

- ステップ 1 に対する手続: クエリ $IV\|m_1$ に対して S は $\mathcal{RO}^\dagger(m_1)$ の出力を用いてレスポンス y_1 を定義して, y_1 を返信する.
- ステップ 2 に対する手続: クエリ $y_1\|m_2$ に対して, y_1 を $\mathcal{TO}$ に質問して m_1 を受け取り, レスポンス y_2 を $chop_n(y_2) = \mathcal{RO}(m_1\|m_2)$ となるように定義する.

この手続きを実行すると, Ideal World で $z = chop_n(y_2)$ となり, さらに $\mathsf{ChopMD}^h \sqsubset_r \mathcal{WRO}$ が証明できる.

これまでの議論では ChopMD^h に対して $\mathcal{O}^*$ を定義していたが, Sponge 構造に対しても同様の議論から $\mathcal{O}^*$ を定義できる. Sponge 構造に対しては, 理想的なブロック暗号 IC と $\mathcal{TO}$ を追加すると上記の 2 つのケースを実現するステイトレスなシミュレータを構成することができて, $\mathsf{Sponge} \sqsubset_r \mathcal{WRO}$ を証明することができる.

以上より, $\mathcal{O}^* := (\mathcal{RO}^*, \mathcal{RO}^\dagger, \mathcal{TO}, \mathsf{IC})$ として, $\mathcal{WRO}$ を $(\mathcal{RO}, \mathcal{RO}^*, \mathcal{RO}^\dagger, \mathcal{TO}, \mathsf{IC})$ から構成して, $\mathcal{WRO}.hon := \mathcal{RO}$, $\mathcal{WRO}.adv := (\mathcal{RO}, \mathcal{RO}^*, \mathcal{RO}^\dagger, \mathcal{TO}, \mathsf{IC})$ とする.

4.1.5.2 $\mathcal{WRO}$ の正しさについて.

上記の議論から $\mathcal{WRO}$ を用いることで, RSS の Reset Indifferentiability に対する不可能性を回避できることを示した. 次に, 重要な $\mathcal{G}_m$ と実用的な $\mathcal{C}$ に対して, $\mathcal{C}^{\mathcal{WRO}}$ の $\mathcal{G}_m$ 安全性を証明して, $\mathcal{WRO}$ 証明法の妥当性を示す.

我々は $\mathcal{G}_m$ として deterministic PKE (DPKE) [3, 5, 18, 31, 43], hedged PKE (HPKE) [4], message-locked PKE (MLPKE) [6]) など多くの公開鍵の安全性として採用されている CDA 安全性を考える. ここで, CDA 安全性はもし攻撃者がチャレンジ暗号文のメッセージと乱数の空間をコントロールできる場合でもメッセージのプライバシーを保証するための安全性である.

次に, $\mathcal{C}$ として REwH1 に注目する. ここで, REwH1 を用いると, 任意の $\mathcal{RO}$ モデルで CPA 安全な公開界鍵暗号アルゴリズムを $\mathcal{RO}$ モデルで IND-SIM 安全な公開鍵暗号アルゴリズムに変換できることが知られている暗号アルゴリズムである [51]. ここで, IND-SIM 安全性は識別者が選んだメッセージと乱数の暗号文とシミュレータが生成した値を識別できないことを保

証した安全性である.

そして, 我々は任意の $\mathcal{RO}$ モデルで IND-SIM 安全な公開鍵暗号アルゴリズムが $\mathcal{WRO}$ モデルで CDA 安全となることを証明する. よって, 既存の REwH1 の結果と合わせると, REwH1 を用いることで $\mathcal{RO}$ モデルで CPA 安全な任意の公開鍵暗号アルゴリズムから $\mathcal{WRO}$ モデルで CDA 安全な公開鍵暗号を得ることができる. これまで多くの $\mathcal{RO}$ モデルで CPA 安全な公開鍵暗号が提案されており, これらを REwH1 に組み込むことで多くの $\mathcal{WRO}$ モデルで CDA 安全な公開鍵暗号を得ることができる.

REwH1 の証明の直観的な説明. 任意の $\mathcal{RO}$ モデルで IND-SIM 安全な公開鍵暗号アルゴリズムが $\mathcal{WRO}$ モデルで CDA 安全となることを証明するために, 追加されたオラクル $\mathcal{O}^*$ が攻撃者に何もアドバンテージを与えないことを示す必要がある. 以下, $\mathcal{O}^*$ が攻撃者にアドバンテージを与えなことを簡単に説明する.

ここで, CDA ゲームは 2 つのステージから構成されるゲームで, 第 1 ステージの攻撃者 $\mathcal{A}_1$ は $m_i \| r$ が十分な min-entropy を持つ 2 つのメッセージ (m_0, m_1) と乱数 r を出力して, 1 bit の値 β をランダムに振って, チャレンジ暗号文 $c_\beta = \mathcal{E}(m_\beta; r)$ を生成して, 第 2 ステージの攻撃者 $\mathcal{A}_2$ は c_β を受け取り, $\mathcal{A}_2$ は β の値を推測する. ここで, $\mathcal{A}_1$ のステイトは $\mathcal{A}_2$ に引き継がれないことに注意されたい.

$\mathcal{WRO}$ は $\mathcal{RO}, \mathcal{RO}^*, \mathcal{RO}^\dagger, \mathcal{TO}$, IC から構成されるため, それぞれのオラクルについて, 攻撃者にアドバンテージを与えないことを説明する. まず, $\mathcal{RO}$ について, IND-SIM 安全性の性質から, $\mathcal{A}_2$ は $\mathcal{RO}$ にアクセスしたとしても c_β から (m_0, m_1) と r の情報を何も得られないことが言える. 次に, $\mathcal{RO}^\dagger$ と $\mathcal{RO}^*$ は (m_0, m_1) と r とは独立のランダムオラクルなので, $\mathcal{RO}^\dagger$ と $\mathcal{RO}^*$ から (m_0, m_1) と r の情報は漏れない. 最後に, IC と $\mathcal{TO}$ からも (m_0, m_1) と r の情報が漏れないことを説明する. ここで, IC の暗号化オラクルを E, 復号オラクルを D と書く. もし, $\mathcal{A}_2$ が (m_0, m_1) と r の情報を D, E または $\mathcal{TO}$ から得られた場合, CDA ゲームに勝つことができる可能性がある. しかし, 次の 2 つの理由から無視できる確率を除いてこのことは起きないことを説明する. 1) $\mathcal{A}_2$ は $\mathcal{A}_1$ が出力する任意の情報は上記の議論から c_β, $\mathcal{RO}$, $\mathcal{RO}^\dagger$, $\mathcal{RO}^*$ から得られない. 2) $\mathcal{RO}^\dagger$, E, D の出力はランダムに選ばれるため, $E, D, \mathcal{TO}$ から情報を得るためにはこのランダムな値を推定するしかない. よって, $\mathcal{TO}$, IC は無視できる確率を除いて攻撃者にアドバンテージを与えない. 以上より, $\mathcal{O}^*$ は攻撃者にアドバンテージを与えず, 結果として, 任意の $\mathcal{RO}$ モデルで IND-SIM 安全な公開鍵暗号アルゴリズムが $\mathcal{WRO}$ モデルで CDA 安全となることを証明することができる.

4.2 Reset Indifferentiability を用いた $\mathcal{WRO}$ 証明法

文献 [51] では ChopMD 構造と Sponge 構造に対して, 部品にランダムオラクル圧縮関数 h とランダム置換 P を組み込んだとしても, ChopMD^h と Sponge^P が $\mathcal{RO}$ と Reset Indifferentiable となることを証明することは不可能であることが示された.

我々はマルチステージゲームで $\mathsf{ChopMD}^{\mathsf{P}}$ と Sponge^P への置き換えを保障するために, $\mathcal{RO}$ を弱めた weakened $\mathcal{RO}$ ($\mathcal{WRO}$ と書く) を提案して, $\mathcal{WRO}$ から ChopMD^h と (出力長が r bit で固定の)Sponge^P との置き換えができることを Reset Indifferentiability を用いて証明する.

まず, $\mathcal{WRO}$ を定義する.

定義 23 $\mathcal{RO}_n$ を入力長が任意長で出力長が n bit のランダムオラクルとする. $\mathcal{RO}_v^*$ を入力長が任意長で出力長が v bit のランダムオラクルとする. $\mathcal{RO}_w^\dagger$ を入力長が任意長で出力長が w bit のランダムオラクルとする. ここで, $\mathcal{RO}_w^\dagger$ のハッシュリストを $\mathcal{L}^\dagger$ と書くことにする. オラクル $\mathcal{TO}$ を次のように定義する.

- $\mathcal{TO}$ への w bit のクエリ z に対して, 以下の手続きを行う
  - **if** $\exists_1(m, z) \in \mathcal{L}^\dagger$ **then return** m
  - **else return** $\perp$

$\mathsf{IC}_{a,b}$ を鍵長が a bit, ブロック長が b bit の理想的なブロック暗号とする. そして, $\mathcal{WRO} := (\mathcal{RO}_n, \mathcal{RO}_v^*, \mathcal{RO}_w^\dagger, \mathcal{TO}, \mathsf{IC}_{a,b})$ とする. ここで, $\mathcal{WRO}.hon := \mathcal{RO}_n$, $\mathcal{WRO}.adv := (\mathcal{RO}_v^*, \mathcal{RO}_w^\dagger, \mathcal{TO}, \mathsf{IC}_{a,b})$ とする. ♦

ここで, 上記のパラメータ (n, v, w, a, b) は固定値ではなく, 証明したいハッシュ関数ごとに値が変わることに注意されたい.

そして, プリミティブ P が理想化されたハッシュ関数 H^{P} に対する Reset Indifferentiable from $\mathcal{WRO}$ のアドバンテージを以下に定義する.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{H}^{\mathsf{P}},\mathcal{WRO},S}(\mathcal{D}) := \Pr[\mathcal{D}^{\mathsf{H}^{\mathsf{P}},\mathsf{P}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{WRO}.hon, S^{\mathcal{WRO}.adv}} \Rightarrow 1].$$

そして, Reset Indifferentiability の定理 (定理 2.6) から, 以下の系が成り立つ.

系 2 $\mathsf{H}^{\mathsf{P}} \sqsubset_r \mathcal{WRO} \Leftrightarrow \forall \mathcal{G} \in \mathbf{G}_m, \forall \mathcal{C} \in \mathbf{C} : \mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ \mathcal{C}^{\mathcal{WRO}}$

4.2.1 ChopMD^h に対する Reset Indifferentiability from $\mathcal{WRO}$

ここでは, h がランダムオラクル圧縮関数の時に, ChopMD^h が $\mathcal{WRO}$ と Reset Indifferentiable となることを証明する. ここで用いる $\mathcal{WRO}$ のパラメータを $w = s$, $v = n + s$ とする. また, IC$_{a,b}$ はこの証明では使われないので省略する.

定理 4.1 任意の識別者 $\mathcal{D}$ に対して, 以下の式を満たすステイトレスなシミュレータ S が存在する.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{ChopMD}^h,\mathcal{WRO},S}(\mathcal{D}) \leq \frac{q_R(q_R-1)+2\sigma(\sigma+1)}{2^s}$$

ここで, Real World は $(L,R) = (\mathsf{ChopMD}^h, h)$ で, Ideal World は $(L,R) = (\mathcal{WRO}.hon, S^{\mathcal{WRO}.adv})$ である. そして, $\mathcal{D}$ が L, R にクエリする回数の最大値を q_L, q_R として, l を 1 ブロックを t bit としたときの 1 回の L クエリで可能なブロック数の最大値とする. そして, $\sigma = lq_L + q_R$ とする. この時, S のクエリ回数は高々 $4q_R$ で, 実行時間は $\mathcal{O}(q_R)$ となる. ♦

4.3 章でこの証明を行う.

4.2.2 Sponge^P に対する Reset Indifferentiability from $\mathcal{WRO}$

この証明では, Sponge^P の出力長を 1 ブロックすなわち r bit に限定する. SHA-3 で用いられている出力長は 1 ブロック分だけなので, この証明で十分に重要なハッシュ関数をカバーできていることに注意されたい. また, $\mathcal{WRO}$ のパラメータを $w = c$, $b = r$ とする. この証明では, IC$_{a,b}$ の鍵を固定値 k^* に固定して使うため a の長さは無視する. そして, $E(k^*,\cdot)$ はランダム置換となるので, $E(k^*,\cdot)$ をランダム置換 $\mathcal{P}$ と書いて, $D(k^*,\cdot)$ を $\mathcal{P}^{-1}$ と書くことにする. また, この証明では $\mathcal{RO}^*_v$ は使われないよって, $\mathcal{WRO} = (\mathcal{RO}_n, \mathcal{RO}^\dagger_c, \mathcal{TO}, \mathcal{P}, \mathcal{P}^{-1})$ となる.

定理 4.2 P をランダム置換とする. P^{-1} をその逆演算を行うオラクルとする. 任意の識別者 $\mathcal{D}$ に対して, 以下の式を満たすシミュレータ $S = (S_F, S_I)$ が存在する.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{Sponge}^P,\mathcal{WRO},S}(\mathcal{D}) \leq \frac{2\sigma(\sigma+1)+q(q-1)}{2^c} + \frac{\sigma(\sigma-1)+q(q-1)}{2^{d+1}}$$

ここで, $\mathcal{D}$ は L オラクル $L = \mathsf{Sponge}^P/\mathcal{RO}_n$ R オラクル $R_F = P/S_F, R_I = P^{-1}/S_I$ にそれぞれ q_L, q_F, q_I 回のクエリを行う. そして, l は 1 回の L へのクエリの最大ブロック長とする. $\sigma = lq_L + q_F + q_I$, $q = q_F + q_I$ とする. S のクエリ回数は $4q$ 回で, 動作時間は $\mathcal{O}(q)$ である. ♦

$\underline{S(x,m)}$ where $x_1 = x[1,s], x_2 = x[s+1,n]$ and $|m| = t$
1 $M \leftarrow \mathcal{TO}(x_1)$;
2 if $x = IV$ then $z \leftarrow \mathcal{RO}_{n-s}(m)$; $w \leftarrow \mathcal{RO}_s^{\dagger}(m)$;
3 else if $M \neq \perp$ and $x_2 = \mathcal{RO}_{n-s}(M)$ then $z \leftarrow \mathcal{RO}_{n-s}(M\|m)$; $w \leftarrow \mathcal{RO}_s^{\dagger}(M\|m)$;
4 else $w\|z \leftarrow \mathcal{RO}_n^*(x\|m)$;
5 **return** $w\|z$;

図.4.3　Simulator S

この証明は 4.4 章で行う.

4.3　定理 4.1 の証明

証明の準備

まず, グラフ G_{MD} を定義する. このグラフの初期値はノード IV で初期化されている. そして, このグラフは R オラクルへのクエリレスポンスから, ノードとエッジが定義される. 例えば, R へのクエリレスポンスとして $(IV, m_1, y_1), (y_1, m_2, y_2)$ が定義されていた場合, すなわち, $y_1 = (IV, m_1), y_2 = R(y_1, m_2)$ の場合, y_1 をノードとし, m_1 を IV から y_1 へのエッジとする, また, y_2 をノードとして, m_2 を y_1 から y_2 へのエッジとする. ここで, IV から y_2 へのグラフを $IV \xrightarrow{m_1} y_1 \xrightarrow{m_2} y_2$ または $IV \xrightarrow{m_1\|m_2} y_2$ と書くことにする.

ここで, 証明を単純にするために, 証明で考える ChopMD^h はパディング関数 pad を取り除いたものを考える. そのため, $\mathcal{D}$ から L へのクエリは $(\{0,1\}^t)^*$ に入っている値とする. pad は $\{0,1\}^*$ から $(\{0,1\}^t)^*$ への単射関数なので, pad を取り除いた ChopMD^h の安全性は pad を含むフルスペックの ChopMD^h の安全性を保証することに注意されたい.

ステイトレスなシミュレータの定義.

まず, 図 4.3 にステイトレスなシミュレータを定義する. 4.1.5.1 章で説明した通り, S が実現すべきケースは 2 つあり, 実現すべきケース 1 は h 自身をシミュレートすること, そして, 実現すべきケース 2 は L と R の関係のシミュレートである.

まず, ケース 1 について, h はランダムオラクル圧縮関数なので, S の出力に対する要求として乱数であることが要求されるが, $\mathcal{RO}_{n-s}, \mathcal{RO}_n^*, \mathcal{RO}_s^{\dagger}$ を用いて S の出力を定義しているた

め, S の出力は乱数となる. また, S の手続きは $\mathcal{TO}$ の出力によって変わってくるため, 同じクエリに対してレスポンスの値が変わる可能性があるが, 無視できる確率を除いて毎回同じ値が返ってくることを示す.

そして, ケース 2 はグラフ G_{MD} を用いると, Real World では任意のグラフ $IV \xrightarrow{m} y$ に対して $y[s+1,n] = \mathsf{ChopMD}^h(m)$ となるため, 同様のことが Ideal World でも成り立つことが要求される. ここで, S へ (IV, m_1) がクエリされると, S のステップ 2 で出力 y_1 が $y_1[s+1,n] = \mathcal{RO}_{n-s}(m_1)$, $y_1[1,s] = \mathcal{RO}^{\dagger}_s(m_1)$ により定義される. そして, $IV \xrightarrow{m_1} y_1$ に対して, $y_1[s+1,n] = \mathcal{RO}_{n-s}(m)$ が成り立つ. 次に, (y_1, m_2) が S にクエリされた場合で, $y_1[1,s] = \mathcal{RO}^{\dagger}_s(m_1^*)$ となる m_1^* が m_1 以外に存在しない場合, ステップ 1 の $\mathcal{TO}$ の出力値は m_1 となる. そして, $y_1 \neq IV$ の場合, ステップ 3 によって, $S(y_1, m_2)$ の出力値 y_2 が $y_2[s+1,n] = \mathcal{RO}_{n-s}(m_1 \| m_2)$, $y_2[1,s] = \mathcal{RO}^{\dagger}_s(m_1 \| m_2)$ により定義される. この議論を続けると, S のクエリレスポンス $(IV, m_1, y_1), (y_1, m_2, y_2), \ldots, (y_{i-1}, m_i, y_i)$ に対して, $IV \xrightarrow{m_1 \| m_2 \| \cdots \| m_i} y_i$ と書けて, このクエリレスポンスが $1, 2, \ldots, i$ の順番で定義されて, $y_j \neq IV$ で $\mathcal{RO}^{\dagger}_s$ の出力に衝突が起こらなければ, $y_i = \mathcal{RO}_{n-s}(m_1 \| m_2 \| \cdots \| m_i)$ となる. すなわち, この場合は, ケース 2 を実現することができる. 一方で, クエリレスポンス $(IV, m_1, y_1), (y_1, m_2, y_2), \ldots, (y_{i-1}, m_i, y_i)$ に対して, このクエリレスポンスが $1, 2, \ldots, i$ の順番で定義されない, ある j について $y_j = IV$ となる, または $\mathcal{RO}^{\dagger}_s$ の出力に衝突が起こる場合に, $y_i \neq \mathcal{RO}_{n-s}(m_1 \| m_2 \| \cdots \| m_i)$ となりケース 2 は実現できないがこれは無視できるくらい小さい確率であることを証明する.

証明の詳細.

以下, 図 4.3 の S に対して, $\mathcal{D}$ は無視できる確率を除き Real World と Ideal World を識別できないことを証明する. この証明では, 以下の 5 つのゲームを考える. 各々のゲームでは, 識別者がアクセスするオラクル (L, R) を変更している.

- Game 1 は Ideal World と同じである. すなわち, $(L, R) = (\mathcal{RO}_n, S)$ となる.
- Game 2 は S を S_1 に変更した $(L, R) = (\mathcal{RO}_n, S_1)$ を考える. ここで S_1 は S_1 のクエリレスポンスペアをテーブル $\mathcal{T}$ に記録しておき, S_1 へのクエリ (x, m) に対して, $(x, m, w \| z) \in \mathcal{T}$ ならば $w \| z$ を返し, そうでなければ $S(x, m)$ と同じ手続きを実行し, この出力値を返す.
- Game 3 は L を $\mathcal{RO}$ から L_1 に変更した $(L, R) = (L_1, S_1)$ を考える. ここで, L_1 はクエリ m に対して, S_1 にアクセスして $\mathsf{ChopMD}^{S_1}(m)$ を計算し, $\mathcal{RO}_n(m)$ を返すオラク

ルである.

- Game 4 は $(L, R) = (\mathsf{ChopMD}^{S_1}, S_1)$ とする.
- Game 5 は Real World と同じである. すなわち, $(L, R) = (\mathsf{ChopMD}^{h}, h)$ となる.

そして, G_i を $\mathcal{D}$ が Game i で 1 を出力するイベントとすると, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{ChopMD}^h, \mathcal{WRO}, S}(\mathcal{D}) \leq \sum_{i=1}^{4} |\Pr[G_i] - \Pr[G_{i+1}]|$$

以下, $i = 1, \ldots, 4$ に対して, $|\Pr[G_i] - \Pr[G_{i+1}]|$ を評価する.

Game 1 ⇒ Game 2. まず, $|\Pr[G_1] - \Pr[G_2]|$ を評価する. Game 1 から Game 2 への変更は R が S から S_1 に変更されている点である. ここで, S_1 はテーブル $\mathcal{T}$ を参照して繰り返しのクエリに対しては過去に返したレスポンスを返すように変更されている. よって, S も同様に繰り返しのクエリに対して, 過去に返したレスポンスを返すことが示せれば, Game 1 と Game 2 は同じゲームとして扱うことができるため, $|\Pr[G_1] - \Pr[G_2]|$ は, Game 1 で S への繰り返しクエリに対して, 過去に返した値とは別の値が返信される確率で抑えることができる. ところで, 図 4.3 の S の仕様から, S の出力値はステップ 1 の $\mathcal{TO}(x_1)$ の出力値によって出力が定義されるステップが決まるため, 任意のクエリ (x, m) に対して, (x, m) が繰り返し S にクエリされても, クエリ毎にステップ 1 の $\mathcal{TO}(x_1)$ の出力値が常に同じならば, このレスポンスは常に同じ手続きで定義されて同じ値が返る. 以上より, $|\Pr[G_1] - \Pr[G_2]|$ は, ある繰り返し行われたクエリのペア (x, m) に対して, (x, m) があるタイミングで S にクエリされた時に, ステップ 1 で $\mathcal{TO}(x_1)$ の出力として w が出力されて, 繰り返し (x, m) がクエリされた時に, ステップ 1 で $\mathcal{TO}(x_1)$ の出力として w とは異なる w^* が出力される確率で抑えることができる. このイベントを **Diff** と書くことにする.

次に, **Diff** を以下の 2 つのイベントに分ける: $\mathbf{Diff}_1$ は $w = \perp$ かつ $w^* \neq \perp$ となるイベントで, $\mathbf{Diff}_2$ は $w \neq \perp$ かつ $w^* = \perp$ となるイベント. この場合以下の式が成り立つ.

$$|\Pr[G_1] - \Pr[G_2]| \leq \Pr[\mathbf{Diff}] = \Pr[\mathbf{Diff}_1] + \Pr[\mathbf{Diff}_2]$$

以下, $\Pr[\mathbf{Diff}_1]$ と $\Pr[\mathbf{Diff}_2]$ を評価する.

まず, $\Pr[\mathbf{Diff}_1]$ を評価する. このイベントでは, 最初の S へのクエリ (x, m) に対して, ステップ 1 の出力は $w = \perp$ となる. すなわち, 最初のクエリが行われた時点では, $x_1 = \mathcal{RO}^{\dagger}_s(m)$ となる m は定義されておらず, 繰り返しクエリ (x, m) で, ステップ 1 の出力は $w^* \neq \perp$ となるため, この時点では, $x_1 = \mathcal{RO}^{\dagger}_s(m)$ となる m が定義されていることになる. ここで, $w^* = m$ であることに注意されたい. すなわち, $\mathbf{Diff}_1$ はある値 x_1 が先に決まっていて $\mathcal{RO}$ の出力値

が偶然 x_1 と等しくなるイベントである. ここで, $\mathcal{RO}_s^\dagger$ と $\mathcal{TO}$ へは高々 q_R 回クエリされるため, 以下の式が成り立つ.

$$\Pr[\mathbf{Diff}_1] \leq \sum_{i=1}^{q_R} \frac{i-1}{2^s} \leq \frac{q_R(q_R-1)}{2^{s+1}}.$$

次に, $\Pr[\mathbf{Diff}_2]$ を評価する. このイベントでは, 最初の S へのクエリ (x, m) に対して, ステップ 1 の出力は $w \neq \perp$ となる. すなわち, 最初のクエリが行われた時点では, $x_1 = \mathcal{RO}_s^\dagger(m)$ となる m が定義されていることになる. ここで, $w^* = m$ であることに注意されたい. そして, 繰り返しクエリ (x, m) で, ステップ 1 の出力は $w^* = \perp$ となるため, $\mathcal{TO}$ の定義から, この時点で $x_1 = \mathcal{RO}_s^\dagger(m^*)$ となる m とは異なる m^* が定義されたことになる. すなわち, $\mathbf{Diff}_2$ が起こると $\mathcal{RO}_s^\dagger$ の出力に衝突が起こる. ここで, $\mathcal{RO}_s^\dagger$ へは高々 q_R 回クエリされるため, 以下の式が成り立つ.

$$\Pr[\mathbf{Diff}_2] \leq \sum_{i=1}^{q_R} \frac{i-1}{2^s} \leq \frac{q_R(q_R-1)}{2^{s+1}}.$$

以上より, 以下の式が成り立つ.

$$|\Pr[G_1] - \Pr[G_2]| \leq \frac{q_R(q_R-1)}{2^s}$$

Game 2 ⇒ Game 3. 次に, $|\Pr[G_2] - \Pr[G_3]|$ を評価する. Game 2 から Game 3 への変更点は, L が $\mathcal{RO}_n$ から L_1 に変更されている点である. ここで, L_1 はクエリ m に対して, $\mathsf{ChopMD}^{S_1}(m)$ を計算するためのクエリを S_1 にしてから $\mathcal{RO}_n(m)$ の出力値を返信するため, $\mathsf{ChopMD}^{S_1}(m)$ によるクエリが $\mathcal{D}$ の振る舞いに影響を与えないことが示せると, Game 2 と Game 3 は同じイベントとして扱うことができる. すなわち, $|\Pr[G_2] - \Pr[G_3]|$ は $\mathsf{ChopMD}^{S_1}(m)$ によるクエリが $\mathcal{D}$ の振る舞いに影響を与える確率で抑えることができる.

ここで, Game 3 の $\mathsf{ChopMD}^{S_1}(m)$ による追加のクエリレスポンスを $m := m_1 \| \cdots \| m_i$ の場合, $(IV, m_1, y_1), (y_1, m_2, y_2), \ldots, (y_{i-1}, m_i, y_i)$ として, グラフ G_{MD} を用いて表現すると, $IV \xrightarrow{m} y_i$ となる. ここで, $y_i[s+1, n] = \mathsf{ChopMD}^{S_1}(m)$ である. 次に, Game i での任意のグラフ $IV \xrightarrow{m} y$ に対して, イベント Bad_i が起こらなければ $y = \mathcal{RO}_s^\dagger(m) \| \mathcal{RO}_n(m)$ となることを補助定理 11 で証明する. イベント Bad_i は補助定理で示す. これは, Bad_2 と Bad_3 が起こらなければ, Game 2, Game 3 ともに, 任意のグラフ $IV \xrightarrow{m} y$ に関する出力値は $y = \mathcal{RO}_s^\dagger(m) \| \mathcal{RO}_n(m)$ となるため, Game 3 の L_1 による追加のクエリは影響を与えない. よって, 以下の式が成り立つ.

$$|\Pr[G_2] - \Pr[G_3]| \leq \max\{\mathsf{Bad}_2, \mathsf{Bad}_3\}$$

補題 11 Game j で任意のグラフ $IV \xrightarrow{m} y$ に対して, 以下のイベント Bad_i が起こらなければ $y = \mathcal{RO}_s^\dagger(m)\|\mathcal{RO}_n(m)$ となる. ここで, S が i 回呼び出された後の S の全てのクエリレスポンスの値 $(x_\beta, m_\beta, y_\beta)$ $(\beta = 1, \ldots, i)$ に対して, $(x_\beta[1,s], y_\beta[1,s])$ を記録したテーブルを T_i と書くことにする.

- Bad_j: Game j で, ある S への i 番目のクエリ (x_i, m_i) のレスポンスが $T_{i-1} \cup \{x_i[1,s], IV[1,s]\}$ の要素と一致する.

証明. Bad_j が起こらないと仮定する. 以下, 任意のグラフ $IV \xrightarrow{m} y$ に対して, $y = \mathcal{RO}_s^\dagger(m)\|\mathcal{RO}_n(m)$ となることを示す. ここで, $(x_1, m_1, y_1), \ldots, (x_t, m_t, y_t)$ をこのグラフに関係する S のクエリレスポンス値とする. $x_1 = IV$, $x_l = y_{l-1}$ $(l = 2, \ldots, t)$, $y_t = y$, and $m = m_1\| \ldots \|m_t$ である.

まず, $t = 1$ と $t \geq 2$ のケースに分けて議論する.

$t = 1$ の場合, S のステップ 2 から $y = \mathcal{RO}_s^\dagger(m_1)\|\mathcal{RO}_n(m_1)$ となる.

次に, $t \geq 2$ の場合を考える. まず, Bad_j が起こらないので, $\forall l \in \{1, \ldots, t-1\}$ に対して, (x, m_i, y_i) が定義された後に, $(x_{i+1}, m_{i+1}, y_{i+1})$ が定義される. すなわち, $(x_1, m_1, y_1), \ldots, (x_t, m_t, y_t)$ の順番で定義される. そして, ステップ 1 から $y_1 = \mathcal{RO}_s^\dagger(m_1)\|\mathcal{RO}_n(m_1)$ となる. 次に, S へのクエリ (y_1, m_2) に対して, $\mathcal{RO}_s^\dagger$ の出力の衝突は起こらないので, S のステップ 1 で m_1 が $\mathcal{TO}$ から出力される. そして, Bad_j が起こらないので, $y_1 \neq IV$ となり, ステップ 3 が実行されて $y_2 = \mathcal{RO}_s^\dagger(m_1\|m_2)\|\mathcal{RO}_n(m_1\|m_2)$ となる. $(x_3, m_3, y_3), \ldots, (x_t, m_t, y_t)$ についても同様の議論が成り立つため, $y_t = \mathcal{RO}_s^\dagger(m_1\| \cdots \|m_t)\|\mathcal{RO}_n(m_1\| \cdots \|m_t)$ となる.

■

次に, $\Pr[\mathsf{Bad}_2]$ と $\Pr[\mathsf{Bad}_3]$ を評価する. S の出力は $\{0,1\}^n$ からランダムに選ばれるため, S への i 番目のクエリ (x_i, m_i) の出力 y_i に対して, $y_i[1,s]$ が $T_i \cup \{x_i[1,s]\} \cup \{IV[1,s]\}$ の要素と衝突する確率は $(2i+1)/2^s$ となる. そして, Game 2 では S は q_R 回, Game 3 では σ 回呼び出されるため, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad}_2] \leq \sum_{i=1}^{q_R} \frac{2(i-1)+2}{2^s} = \frac{q_R(q_R+1)}{2^s},\ \Pr[\mathsf{Bad}_3] \leq \sum_{i=1}^{\sigma} \frac{2(i-1)+2}{2^s} = \frac{\sigma(\sigma+1)}{2^s}$$

以上より, 以下の式が成り立つ.

$$|\Pr[G_2] - \Pr[G_3]| \leq \frac{\sigma(\sigma+1)}{2^s}.$$

Game 3 ⇒ Game 4. 次に, $|\Pr[G_3] - \Pr[G_4]|$ を評価する. Game 3 から Game 4 への変更点は, L が L_1 から ChopMD^{S_1} に変更されている点である. ここで, Game 3 では任意の L へのクエリ m に対して $\mathcal{RO}(m)$ の出力値がレスポンスとなるので, 任意の L へのクエリ m に対して $\mathsf{ChopMD}^{S_1}(m) = \mathcal{RO}(m)$ となれば Game 3 と Game 4 は同じゲームとして扱うことができる. すなわち, $|\Pr[G_3] - \Pr[G_4]|$ は Game 4 である L へのクエリ m に対して, $\mathsf{ChopMD}^{S_1}(m) \neq \mathcal{RO}(m)$ となる確率で抑えることができる. ここで, 補助定理 11 から, Bad_4 が起こらなければ $\mathsf{ChopMD}^{S_1}(m) = \mathcal{RO}(m)$ となるため, 以下の式が成り立つ.

$$|\Pr[G_3] - \Pr[G_4]| \leq \Pr[\mathsf{Bad}_4]$$

そして, S の出力値は $\{0,1\}^n$ からランダムに選ばれるので, S への i 番目のクエリ (x_i, m_i) の出力 y_i に対して, $y_i[1,s]$ が $T_i \cup \{x_i[1,s]\} \cup \{IV[1,s]\}$ の要素と衝突する確率は $(2i+1)/2^s$ となる. そして, Game 4 では σ 回呼び出されるため, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad}_4] \leq \sum_{i=1}^{\sigma} \frac{2(i-1)+2}{2^s} = \frac{\sigma(\sigma+1)}{2^s}$$

Game 4 ⇒ Game 5. 最後に, $|\Pr[G_4] - \Pr[G_5]|$ を評価する. Game 4 から Game 5 への変更点は, R が S_1 から h に変更されている点である. ここで, S_1 と h レスポンスはともに $\{0,1\}^n$ からランダムに選ばれるため, $\Pr[G_4] = \Pr[G_5]$ となる.

以上より, 定理が証明できた.

■

4.4 定理 4.2 の証明

証明の準備

定理 4.1 の証明と同様に, まずは Sponge 構造に関するグラフ G_S を定義する. このグラフはノード IV で初期化されている. そして, このグラフは R オラクルまたは R^{-1} オラクルへのクエリレスポンスからノードとエッジが定義される. 例えば, R または R^{-1} へのクエリレスポンスとして, $x_1[r+1,c] = IV_1, y_1[r+1,c] = x_2[r+1,c]$ となる $(x_1, y_1), (x_2, y_2)$ が定義されていた場合, すなわち, $y_1 = R(x_1)$ (または $x_1 = R^{-1}(y_1)$), $y_2 = R(x_2)$ (または $x_2 = R^{-1}(y_2)$) の場合, y_1, y_2 をノードとし, $m_1 = IV_1 \oplus x_1[1,r]$, $m_2 = y_1[1,r] \oplus x_2[1,r]$ をエッジとする. ここで, IV から y_2 へのグラフを $IV \xrightarrow{m_1} y_1 \xrightarrow{m_2} y_2$ または $IV \xrightarrow{m_1 \| m_2} y_2$ と書くことにする.

$\underline{S_F(x)}$ $//x_1 = x[1,r],\ x_2 = x[r+1,r+c]$	$\underline{S_I(y)}$ $//y_1 = y[1,r],\ y_2 = y[r+1,r+c]$
1. $m \leftarrow \mathcal{TO}(x_2)$	1. $m \leftarrow \mathcal{TO}(y_2)$
2. **if** $x_2 = IV_2$ **then**	2. **if** $m \neq \perp$ and $\|m\| = n$ **then**
3. $\quad y[1,r] \leftarrow \mathcal{RO}_r(x_1 \oplus IV_1)$	3. $\quad x[1,r] \leftarrow IV_1 \oplus m$
4. $\quad y[r+1,r+c] \leftarrow \mathcal{RO}_c^{\dagger}(x_1 \oplus IV_1)$	4. $\quad x[r+1,r+c] \leftarrow IV_2$
5. **else if** $m \neq \perp$ **then**	5. **if** $m \neq \perp$ and $\|m\| > n$ **then**
6. $\quad m' \leftarrow x[1,r] \oplus \mathcal{RO}_r(m)$	6. $\quad$ let $m = m^*\|\|m'$ $(\|m'\| = r)$
7. $\quad y[1,r] \leftarrow \mathcal{RO}_r(m\|\|m')$	7. $\quad x[1,r] \leftarrow m' \oplus \mathcal{RO}_r(m^*)$
8. $\quad y[r+1,r+c] \leftarrow \mathcal{RO}_c^{\dagger}(m\|\|m')$	8. $\quad x[r+1,r+c] \leftarrow \mathcal{RO}_c^{\dagger}(m^*)$
9. **else** $y \leftarrow \mathcal{P}(x)$	9. else $x \leftarrow \mathcal{P}^{-1}(y)$
10. **return** y	10. **return** x

図.4.4　Simulator (S_F, S_I)

$\underline{\mathcal{P}_1(X)}$	$\underline{\mathcal{P}_1^{-1}(X)}$
1 if $\exists(j,X,Y) \in \mathcal{Q}$ then **return** Y;	1 if $\exists(j,X,Y) \in \mathcal{Q}$ then **return** X;
2 $Y \xleftarrow{\$} \{0,1\}^d$; $\mathcal{Q} \xleftarrow{\cup} (t,X,Y)$; $t \leftarrow t+1$;	2 $X \xleftarrow{\$} \{0,1\}^d$; $\mathcal{Q} \xleftarrow{\cup} (t,X,Y)$; $t \leftarrow t+1$;
3 **return** Y;	3 **return** X;

図.4.5　$\mathcal{Q}$ は初期状態が空集合となっているリストで, t の初期値は $t = 1$ である. $(\mathcal{P}_1, \mathcal{P}_1^{-1})$ のステップ 1 では j が最大値となる 3 つ組に注目する.

ここで, この証明を単純にするために, 証明で扱う Sponge^P は pad 関数を取り除いたものを考える. そのため, $\mathcal{D}$ から L へのクエリは $(\{0,1\}^r)^*$ に入っている値とする. ここで, pad は $\{0,1\}^*$ から $(\{0,1\}^r)^*$ への単射関数なので, pad を取り除いた Sponge^P の安全性は pad を含むフルスペックの Sponge^P の安全性を保証することに注意されたい.

ステイトレスなシミュレータの定義

図 4.4 にステイトレスなシミュレータ $S = (S_F, S_I)$ を定義する. ここで, S_F は P をシミュレートして, S_I は P^{-1} をシミュレートする.

証明の詳細.

以下, 図 4.4 の S に対して, $\mathcal{D}$ は無視できる確率を除き Real World と Ideal World を識別できないことを証明する. この証明では, 以下の 5 つのゲームを考える. 各々のゲームでは, 識別者がアクセスするオラクル (L, R_F, R_I) を変更している.

- Game 1 は Ideal World と同じである. すなわち, $(L, R_F, R_I) = (\mathcal{RO}_n, S_F, S_I)$ となる.
- Game 2 では (P, P^{-1}) を図 4.5 の (P_1, P_1^{-1}) に変更する. そして, S_F, S_I は (P, P^{-1}) の代わりに (P_1, P_1^{-1}) にアクセスする.
- Game 3 では R_F, R_I を S_F, S_I から $S1_F, S1_I$ に変更する. すなわち, $(L, R_F, R_I) = (\mathcal{RO}_r, S1_F, S1_I)$ となる. ここで, $S1 = (S1_F, S1_I)$ とする. そして, $S1$ は $S1_F$ と $S1_I$ のクエリレスポンンスペアをテーブル $\mathcal{T}$ に記録しておき, S_F へのクエリ x に対して, $\exists(x, y) \in \mathcal{T}$ ならば y を返し, そうでなければ $y \leftarrow S_F(x)$ として, (x, y) を $\mathcal{T}$ に記録して, y を返す. 同様に, S_I へのクエリ y に対して, $\exists(x, y) \in \mathcal{T}$ ならば x を返し, そうでなければ $x \leftarrow S_F(y)$ として, (x, y) を $\mathcal{T}$ に記録して, x を返す.
- Game 4 は L を $\mathcal{RO}_r$ から L_1 に変更して, $(L, R_F, R_I) = (L_1, S1_F, S1_I)$ とする. L_1 の定義は, L_1 へのクエリ m に対して, $\mathsf{Sponge}^{S1_F}(m)$ をするために必要なクエリを $S1_F$ に行い, $\mathcal{RO}_r(m)$ を返信する.
- Game 5 は L を L_1 から Sponge^{S1_F} に変更して, $(L, R_F, R_I) = (\mathsf{Sponge}^{S1_F}, S1_F, S1_I)$ とする.
- Game 6 は Real World と同じである. すなわち, $(L, R_F, R_I) = (\mathsf{Sponge}^P, P, P^{-1})$ とする.

ここで, G_i を Game i で $\mathcal{D}$ が 1 を出力するイベントとすると, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{Sponge}^P, \mathcal{WRO}, S}(\mathcal{D}) \leq \sum_{i=1}^{5} |\Pr[G_i] - \Pr[G_{i+1}]|$$

以下, $i = 1, \ldots, 5$ に対して, $|\Pr[G_i] - \Pr[G_{i+1}]|$ を評価する.

Game 1 ⇒ Game 2. まず, $|\Pr[G_1] - \Pr[G_2]|$ を評価する. Game 1 から Game 2 の変更点は, (R_F, R_I) を (P, P^{-1}) から (P_1, P_1^{-1}) に変更した点である. そして, PRF/PRP switching

lemma [10] (詳しくは付録を参照されたい) を適用すると, 以下の式が成り立つ.

$$|\Pr[G_1] - \Pr[G_2]| \leq \frac{q(q-1)}{2^{d+1}}.$$

Game 2 ⇒ Game 3. 次に, $|\Pr[G_2] - \Pr[G_3]|$ を評価する. Game 2 から Game 3 への変更点は, (R_F, R_I) を (S_F, S_I) から $(S1_F, S1_I)$ に変更した点である. ここで, $S1_F$ と $S1_I$ はクエリレスポンスを記録したテーブル $\mathcal{T}$ を参照して, 繰り返し行われたクエリに対して, 過去に返したレスポンスと同じ値を返すように変更されている. よって, Game 2 で S_F と S_I が繰り返しクエリに対して, 過去に返したレスポンスと同じ値を返すことが示せれば, Game 2 と Game 3 を同じゲームとして扱うことができる. すなわち, $|\Pr[G_2] - \Pr[G_3]|$ は Game 2 で, ある繰り返しクエリに対して, 過去に返したレスポンスと別の値が返される確率で抑えられる.

ここで, 繰り返しクエリは以下の 4 パタンに分けられる.

1. S_F への繰り返しクエリ: (1) x が S_F へクエリされて, y が返される. (2) 次に, 同じ値 x が S_F に繰り返しクエリされて, y^* が返される. 同じ値が返るとは $y = y^*$ となることである.
2. S_I への繰り返しクエリ: (1) y が S_I へクエリされて, x が返される. (2) 次に, 同じ値 y が S_I に繰り返しクエリされて, x^* が返される. 同じ値が返るとは $x = x^*$ となることである.
3. S_F, S_I への繰り返しクエリ: (1) x が S_F へクエリされて, y が返されて, (2) 次に, y が S_I にクエリされて, x^* が返される. 同じ値が返るとは $x = x^*$ となることである.
4. S_I, S_F への繰り返しクエリ: (1) y が S_I へクエリされて, x が返されて, (2) 次に, y が S_F にクエリされて, x^* が返される. 同じ値が返るとは $x = x^*$ となることである.

ここで, 2 つのイベント $\mathbf{Diff}_1, \mathbf{Diff}_2$ を定義して, ある繰り返しクエリに対して, 過去に返したレスポンスと別の値が返されるならば $\mathbf{Diff}_1 \vee \mathbf{Diff}_2$ が起きていることを示す. すなわち, $|\Pr[G_2] - \Pr[G_3]| \leq \Pr[\mathbf{Diff}_1] + \Pr[\mathbf{Diff}_2]$ となる.

まず, $\mathbf{Diff}_1, \mathbf{Diff}_2$ を定義する.

- $\mathbf{Diff}_1$: $\mathcal{RO}_c^\dagger$ の出力に衝突が起こる.
- $\mathbf{Diff}_2$: まず, y が $\mathcal{TO}$ にクエリされて, $\perp$ が返信される. 次に, $\mathcal{RO}_c^\dagger$ に m がクエリされて, y が返される.

まず, パタン 1 で $y \neq y^*$ ならば $\mathsf{Bad}_1 \vee \mathsf{Bad}_2$ が起こることを示す. S_F の出力が定義される手続きは $\mathcal{TO}$ の出力に依存する. $\mathcal{TO}$ の出力が決まると, S_F のステップ 2 以降の手続きは決

定的なので, (1) のクエリ x で, $\mathcal{TO}$ から w が出力されて, (2) のクエリ x で, $\mathcal{TO}$ から w^* が出力されるとすると, $y \neq y^*$ ならば $w \neq w^*$ となる. この場合, w, w^* は, $w = \perp \wedge w^* \neq \perp$ または $w \neq \perp \wedge w^* = \perp$ の 2 パタン取りうる. そして, $w = \perp \wedge w^* \neq \perp$ の場合は Bad_2 そのものである. $w \neq \perp \wedge w^* = \perp$ が起こる場合は, $\mathcal{TO}$ の定義から $\mathcal{RO}_c^\dagger$ の出力に衝突が起きていることから生じるため, Bad_1 が起こる.

次に, パタン 2 で $x \neq x^*$ となる場合は, パタン 1 と同じ議論により $\mathbf{Diff}_1 \vee \mathbf{Diff}_2$ が起こる.

次に, パタン 3 で $x \neq x^*$ ならば $\mathbf{Diff}_1 \vee \mathbf{Diff}_2$ が起こることを示す.

- まず, $x[r+1, r+c] = IV_2$ の場合, $y = \mathcal{RO}_r(m_1) \| \mathcal{RO}_c^\dagger(m_1)$ となる. $m_1 = x[1, r] \oplus IV_1$ である. そして, S_I へのクエリ y に対して, $\mathcal{RO}_c^\dagger$ の出力にコリジョンが起こらなければ, すなわち, Bad_1 が起こらなければ, ステップ 1 の $\mathcal{TO}$ の出力は m_1 となるので, $y^* = \mathcal{RO}_r(m_1) \| \mathcal{RO}_c^\dagger(m_1)$ となり, $x \neq x^*$ となる. すなわち, $x[r+1, r+c] = IV_2$ の場合, $x \neq x^*$ ならば $\mathbf{Diff}_1$ が起こる.
- 次に, (1) のクエリ x に対して, S_F の手続きで, $x[r+1, r+c] \neq IV_2$ で $\mathcal{TO}$ の出力 m_1 が $m_1 \neq \perp$ の場合, $y = \mathcal{RO}_r(m_1 \| m_2) \| \mathcal{RO}_c^\dagger(m_1 \| m_2)$ となる. $m_2 = x[1, r] \oplus \mathcal{RO}_r(m_1)$ である. そして, (2) のクエリ y に対して, $\mathcal{RO}_c^\dagger$ の出力にコリジョンが起こらなければ, すなわち, $\mathbf{Diff}_1$ が起こらなければ, S_I の手続きのステップ 1 で $\mathcal{TO}$ から $m_1 \| m_2$ が返り, $x[1, r] = m_2 \oplus \mathcal{RO}_r(m_1)$, $x[r+1, r+c] = \mathcal{RO}_c^\dagger(m_1)$ となり, $x \neq x^*$ となる. すなわち, この場合, $x \neq x^*$ ならば $\mathbf{Diff}_1$ が起こる.
- 最後に, (1) のクエリ x に対して, S_F の手続きで, $x[r+1, r+c] \neq IV_2$ で $\mathcal{TO}$ の出力 m_1 が $m_1 \neq \perp$ の場合, S_F へのクエリ x に対して, ステップ 1 の $\mathcal{TO}$ の出力は $\perp$ となり, $y = \mathcal{P}(x)$ となる. そして, (2) のクエリ y に対して, $\mathbf{Diff}_1$ が起こらなければ, S_I のステップ 1 で $\perp$ が返り, そして $x^* = \mathcal{P}^{-1}(y)$ となり, $x \neq x^*$ となる. すなわち, この場合, $x \neq x^*$ ならば $\mathbf{Diff}_2$ が起こる.

最後に, パタン 4 で $y \neq y^*$ となる場合は, パタン 3 と同じ議論により $\mathbf{Diff}_1 \vee \mathbf{Diff}_2$ が起こる.

次に, $\Pr[\mathbf{Diff}_1]$ と $\Pr[\mathbf{Diff}_2]$ を評価する. まず, $\Pr[\mathbf{Diff}_1]$ を評価する. ここで, R_F, R_I へは高々 q 回クエリできるため, 以下の式が成り立つ.

$$\Pr[\mathbf{Diff}_1] \leq \sum_{i=1}^{q} \frac{i-1}{2^c} \leq \frac{q(q-1)}{2^{c+1}}.$$

次に, $\Pr[\mathbf{Diff}_2]$ を評価する. ここで, R_F, R_I へは高々 q 回クエリできるため, 以下の式が成り立つ.

$$\Pr[\mathbf{Diff}_2] \leq \sum_{i=1}^{q} \frac{i-1}{2^c} \leq \frac{q(q-1)}{2^{c+1}}.$$

以上より, 以下の式が成り立つ.

$$|\Pr[G_2] - \Pr[G_3]| \leq \Pr[\mathbf{Diff}_1] + \Pr[\mathbf{Diff}_2] \leq \frac{q(q-1)}{2^c}.$$

Game 3 ⇒ Game 4. 次に, $|\Pr[G_3] - \Pr[G_4]|$ を評価する. Game 3 から Game 4 への変更点は, L を $\mathcal{RO}_r$ から L_1 に変更した点である. ここで, L_1 はクエリ m に対して $\mathrm{FOLSponge}^{S1_F}(m)$ を計算するために $S1_F$ に関連したクエリを行う. そのため, この追加で行われるクエリが $\mathcal{D}$ の振る舞いに影響を与えないことが示せれば, Game 3 と Game 4 は同じゲームとして扱うことができる. ここで, 補助定理 12 で, ゲーム j でイベント Bad_j が起こらなければ, 任意のグラフ $IV \xrightarrow{m} y$ に対して, $y = \mathcal{RO}_r(m) \| \mathcal{RO}_c^{\dagger}(m)$ となることを証明する. Bad_j は補助定理 12 で示す. すなわち, $IV \xrightarrow{m} y$ という形で書ける R_F, R_I へのクエリレスポンスは Bad_j が起こらなければ常に同じ方法で定義される. そして, Game 4 で L_1 による追加のクエリレスポンスは $IV \xrightarrow{m} y$ という形で書けるため, Game 3 と Game 4 でこのイベントが起こらなければこの追加の追加のレスポンスがあるなしにかかわらず, 同じゲームとなる. よって, 以下の式が成り立つ.

$$|\Pr[G_3] - \Pr[G_4]| \leq \max\{\Pr[\mathsf{Bad}_3], \Pr[\mathsf{Bad}_4]\}.$$

補題 12 Game j で, Bad_j が起こらなければ, 任意のグラフ $IV \xrightarrow{m} z$ に対して, $z = \mathcal{RO}_r(m) \| \mathcal{RO}_c^{\dagger}(m)$ となる.

以下, Bad_j の定義である. T_i を i 番目の R クエリ時に, すでに定義されている R_F と R_I のクエリレスポンスペア (x, y) に対して, $x[r+1, b], y[r+1, b]$ と, IV_2 を記録したテーブルとする.

- Bad_j は, Game j で, R_F または R_I への i 番目のクエリが
  - R_F へのクエリで, そのクエリを x_i, そのレスポンス y_i と書いた時に, $y_i[r+1, b] \in T_i \cup \{x_i[r+1, b]\}$ となる, または,
  - R_I へのクエリで, そのクエリを y_i, そのレスポンス x_i と書いた時に, $x_i[r+1, b] \in T_i \cup \{y_i[r+1, b]\}$ となる.

証明. Bad_j が起こらないと仮定する. $IV \xrightarrow{m} y$ を任意のグラフとする. そして, $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(t)}, y^{(t)})$ をこのグラフを規定する R_F または R_I へのクエリレスポンスとして, $(x^{(i)}, y^{(i)})$ をこのグラフの i ブロック目のクエリレスポンスとする. すなわち, $x^{(1)}[r+1, b] = IV_2$, $y^{(i)}[r+1, b] = x^{(i+1)}[r+1, b]$, $z = y^{(t)}$ となる. そして, $m_1 = IV_1 \oplus x^{(1)}[1, r]$, $m_i = y^{(i-1)}[1, r] \oplus x^{(i)}[1, r]$ とすると, $m = m_1 \| \cdots \| m_t$ である. $z = \mathcal{RO}_r(m) \| \mathcal{RO}_c^\dagger(m)$ となることが示せれば証明完了となる.

ここで, Bad_j が起こらないので, $\mathcal{RO}_c^\dagger$ の出力に衝突は起きないこと注意されたい.

そして, まず $t = 1$ の場合を考える. Bad_j は起きないので, $(x^{(1)}, y^{(1)})$ は S_I によって定義されない, すなわち, S_F によって」定義される. そして, S_F のステップ 2-4 から $z = \mathcal{RO}_r(m) \| \mathcal{RO}_c^\dagger(m)$ となる.

次に, $t \geq 2$ となる任意の t を考える. ここで, Bad_j は起きないので, $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, ..., $(x^{(t)}, y^{(t)})$ の順番で定義される. そして, 任意の i に対して, $(x^{(i)}, y^{(i)})$ は S_I によって定義されない, すなわち, S_F によって定義される. ここで, $t = 1$ のケースで議論したように, $y^{(1)} = \mathcal{RO}_r(m_1) \| \mathcal{RO}_c^\dagger(m_1)$ となる. そして, R_F へのクエリ $x^{(2)}$ に対して, $\mathcal{RO}_c^\dagger$ の出力に衝突が起こらないので, S_F のステップ 1 の手続きで $\mathcal{TO}$ から m_1 が返り, ステップ 5-8 の手続きにより $y^{(2)} = \mathcal{RO}_r(m_1 \| m_2) \| \mathcal{RO}_c^\dagger(m_1 \| m_2)$ となる. 同じ議論を $(x^{(3)}, y^{(3)})$, ..., $(x^{(t)}, y^{(t)})$ に対しても適用できて, $y^{(t)} = \mathcal{RO}_r(m_1 \| \cdots \| m_t) \| \mathcal{RO}_c^\dagger(m_1 \| \cdots \| m_t)$ となる.

■

次に, $\Pr[\mathsf{Bad}_3]$, $\Pr[\mathsf{Bad}_4]$ を評価する. S_F と S_I の出力はの c bit 部分は $\mathcal{RO}_c^\dagger$, $\mathcal{P}_1$, または $\mathcal{P}_1^{-1}$ によって選ばれる. そして, これらの値はランダムに選ばれる. Game 3 では R_F, R_I 合わせて q 回呼び出されて, Game 4 では R_F, R_I 合わせて σ 回呼び出されるため, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad}_3] \leq \sum_{i=1}^{q} \frac{(2(i-1)+2)}{2^c} = \frac{q(q+1)}{2^c}$$
$$\Pr[\mathsf{Bad}_4] \leq \sum_{i=1}^{\sigma} \frac{(2(i-1)+2)}{2^c} = \frac{\sigma(\sigma+1)}{2^c}$$

よって, 以下の式が成り立つ.

$$|\Pr[G_3] - \Pr[G_4]| \leq \max\{\Pr[\mathsf{Bad}_3], \Pr[\mathsf{Bad}_4]\} \leq \frac{\sigma(\sigma+1)}{2^c}$$

Game 4 ⇒ Game 5. 次に, $|\Pr[G_4] - \Pr[G_5]|$ を評価する. Game 4 から Game 5 への変更

点は L を L_1 から Sponge^{S1_F} に変更した点である．ここで, クエリ m に対して L は $\mathcal{RO}_r(m)$ を返すので, Game 5 でもクエリ m に対して L が $\mathcal{RO}_r(m)$ を返すことが示せれば, Game 4 と Game 5 を同じゲームとして扱うことができる．ここで, 補助定理 12 から Bad_5 が起こらない限り Sponge^{S1_F} の出力は $\mathcal{RO}_r$ によって定義されるため, 以下の式が成り立つ.

$$|\Pr[G_4] - \Pr[G_5]| \leq \Pr[\mathsf{Bad}_5]$$

次に, $\Pr[\mathsf{Bad}_5]$ を評価する．S_F と S_I の出力はの c bit 部分は $\mathcal{RO}_c^{\dagger}$, $\mathcal{P}_1$, または $\mathcal{P}_1^{-1}$ によって選ばれる．そして, これらの値はランダムに選ばれる．Game 4 では R_F, R_I 合わせて σ 回呼び出されるため, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad}_5] \leq \sum_{i=1}^{\sigma} \frac{(2(i-1)+2)}{2^c} = \frac{\sigma(\sigma+1)}{2^c}$$

よって, 以下の式が成り立つ.

$$|\Pr[G_4] - \Pr[G_5]| \leq \max\{\Pr[\mathsf{Bad}_4], \Pr[\mathsf{Bad}_5]\} \leq \frac{\sigma(\sigma+1)}{2^c}$$

Game 5 ⇒ Game 6. 最後に, $|\Pr[G_5] - \Pr[G_6]|$ を評価する．Game 5 から Game 6 への変更点は (R_F, R_I) を $(S1_F, S1_I)$ から (P, P^{-1}) に変更した点である．この変更に対して, PRF/PRP switching lemma [10] (詳しくは付録を参照されたい) が適用できて, 以下の式が成り立つ.

$$|\Pr[G_5] - \Pr[G_6]| \leq \frac{\sigma(\sigma-1)}{2^{b+1}}.$$

以上より, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{r\text{-}indiff}}_{\mathsf{Sponge}^P, \mathcal{WRO}, S}(\mathcal{D}) \leq \frac{2\sigma(\sigma+1)+q(q-1)}{2^c} + \frac{\sigma(\sigma-1)+q(q-1)}{2^{b+1}}$$

■

4.5 $\mathcal{WRO}$ モデルでのマルチステージゲームの安全性について

本章では, $\mathcal{WRO}$ 方法論の正しさを示す．ここでは, $\mathcal{WRO}$ モデルで (non-adaptive) CDA 安全 [4] な公開鍵暗号方式を構成する．この構成法は, $\mathcal{RO}$ モデルで IND-SIM 安全 [51] な公開鍵暗号から $\mathcal{WRO}$ モデルで CDA 安全な公開鍵暗号を構成する方法である.

ここで, 文献 [51] で, EwH [3] や REwH1 [4] を用いると容易に $\mathcal{RO}$ モデルで IND-SIM 安全な公開鍵暗号方式を得ることができることが示されている. 具体的には, EwH と REwH1 に任意の CPA 安全な公開鍵暗号方式を部品として組み込むと, IND-SIM 安全な方式が得られることが文献 [51] で示されている. よって, 我々が示す結果と組み合わせると, 任意の $\mathcal{RO}$ モデルで CPA 安全な公開鍵暗号方式から $\mathcal{WRO}$ モデルで CDA 安全な公開鍵暗号方式を得ることができる. CPA 安全な公開鍵暗号方式として, 例えば, 素因数分解ベースの方式, Diffie-Hellman ベースの方式, 格子ベースの方式など多くの方式があり, これらの方式を組み込むと, 同様の仮定をベースとした CDA 安全な公開鍵暗号方式が得られる.

さらに, 4.2 章での任意のマルチステージゲームで $\mathcal{WRO}$ から ChopMD または Sponge への置き換えを保障した結果と合わせると, 任意の $\mathcal{RO}$ モデルで CPA 安全な公開鍵暗号方式から CDA 安全な ChopMD または Sponge を用いた公開鍵暗号方式を得ることができる.

4.5.1 $\mathcal{WRO}$ モデルで CDA 安全な公開鍵暗号方式

公開鍵暗号方式 (Public Key Encryption または PKE). 公開鍵暗号方式 $\mathcal{AE}$ は 3 つのアルゴリズム $\mathcal{AE} = (\mathsf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ から構成される. Gen は公開鍵 pk と秘密鍵 sk を出力する鍵生成アルゴリズムである. **Enc** は公開鍵 pk, 平文 m, 及び乱数 r を用いて暗号文 c を出力する暗号化アルゴリズムである. **Dec** は秘密鍵 sk と暗号文 c を用いて平文 m またはリジェクトシンボル $\perp$ を出力するアルゴリズムである. ここで, ベクトルサイズが $|\mathbf{m}| = |\mathbf{r}| = l$ となるベクトル $\mathbf{m}, \mathbf{r}$ に対して, ベクトル $(\mathbf{Enc}(pk, \mathbf{m}[1]; \mathbf{r}[1]), \ldots, \mathbf{Enc}(pk, \mathbf{m}[l]; \mathbf{r}[l]))$ を $\mathbf{Enc}(pk, \mathbf{m}; \mathbf{r})$ と書くことにする. ここで, **Enc** が決定的の場合, $\mathcal{AE}$ は決定的であるということにする.

CDA 安全性. ここで, CDA 安全性を説明する. 本章では, 文献 [51] に記載されている定義を用いる. 図 4.7 は暗号部品 F を用いた公開鍵暗号 $\mathcal{AE}$ に対する non-adaptive な CDA ゲームである.

この定義は公開鍵暗号方式で用いる乱数 $\mathbf{r}$ が一様に選ばれない時の安全性を考慮に入れた安全性定義である. 以降では, 乱数の長さを $\rho \geq 0$, 平文の長さを $\omega > 0$ と書くことにする. ここで, (μ, ν)-mmr-source $\mathcal{M}$ は三つ組み $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ を出力するランダマイズアルゴリズムである. ここで, $|\mathbf{m}_0| = |\mathbf{m}_1| = |\mathbf{r}| = \nu$, $\mathbf{m}_0$ と $\mathbf{m}_1$ の全てのコンポーネントは長さが ω bit のビット列で, $\mathbf{r}$ の全てのコンポーネントは長さが ρ bit のビット列で, $1 \leq i < j \leq \nu$, $\beta \in \{0, 1\}$ に対して, $(\mathbf{m}_\beta[i], \mathbf{r}[i]) \neq (\mathbf{m}_\beta[j], \mathbf{r}[j])$ となる.

$\underline{\mathrm{CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathcal{AE},F}}$
$\beta \xleftarrow{\$} \{0,1\}$
$(pk, sk) \xleftarrow{\$} \mathsf{Gen}$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow \mathcal{A}_1^{F.adv}$
$\mathbf{c} \leftarrow \mathbf{Enc}^{F.hon}(pk, \mathbf{m}_\beta; \mathbf{r})$
$\beta' \leftarrow \mathcal{A}_2^{F.adv}(pk, \mathbf{c})$
return $(\beta = \beta')$

図.4.6　IND-SIM ゲーム

$\underline{\text{IND-SIM}^{\mathcal{B}}_{\mathcal{AE},\mathcal{S},F}}$
$\beta \xleftarrow{\$} \{0,1\}$
$(pk, sk) \xleftarrow{\$} \mathsf{Gen}$
$\beta' \leftarrow \mathcal{B}^{\mathsf{RoS},F.adv}(pk)$
return $(\beta = \beta')$

$\underline{\mathsf{RoS}(m, r)}$
If $\beta = 1$ then **return** $\mathbf{Enc}^{F.hon}(pk, m; r)$
Otherwise **return** $\mathcal{S}^{F.hon}(pk, |m|)$

図.4.7　CDA ゲームと IND-SIM ゲーム

IND-SIM 安全性.　IND-SIM 安全性は公開鍵方式の安全性で, 攻撃者が平文や乱数を選んで, そのペアを暗号アルゴリズムに入力したときの出力値とあるシミュレータの出力値を識別できないことを規定した安全性である. 図 4.7 は IND-SIM ゲームを記載している. ここで, 攻撃者 $\mathcal{B}$ の IND-SIM のアドバンテージを以下のように定義する.

$$\mathsf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S},F}(\mathcal{B}) = 2 \cdot \Pr[\text{IND-SIM}^{\mathcal{B}}_{\mathcal{AE},F} \Rightarrow \mathsf{true}] - 1.$$

$\mathcal{WRO}$ モデルでの CDA 安全性. 以下の定理は $\mathcal{RO}$ モデルで IND-SIM 安全な任意の公開鍵から $\mathcal{WRO}$ モデルで non-adaptive CDA 安全な公開鍵を得ることができることを示している.

定理 4.3 $\mathcal{AE}$ を公開鍵暗号とする. $(\mathcal{A}_1, \mathcal{A}_2)$ を $\mathcal{WRO}$ モデルでの CDA 攻撃者として, $\mathcal{RO}_n$, $\mathcal{RO}^*_v$, $\mathcal{RO}^\dagger_w$, $\mathcal{TO}$, $\mathsf{IC}_{a,b} = (E, D)$ へのクエリ回数をそれぞれ $q_{\mathcal{RO}}, q_{\mathcal{RO}^*}, q_{\mathcal{RO}^\dagger}, q_{\mathcal{TO}}, q_E, q_D$ とする. そして, 任意のシミュレータ $\mathcal{S}$ に対して, 以下の式を満たす IND-SIM 攻撃者 $\mathcal{B}$ が存

在する.

$$\mathsf{Adv}^{\mathrm{cda}}_{\mathcal{AE},\mathcal{WRO}}(\mathcal{A}_1,\mathcal{A}_2) \le \mathsf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S},\mathcal{RO}_n}(\mathcal{B}) + q_{\mathcal{RO}} \cdot \mathsf{maxpk}_{\mathcal{AE}} + \frac{q_{\mathcal{RO}} + 4q^2_{\mathcal{RO}^*}}{2^\mu}$$
$$+ \max\left\{\frac{4q^2_{\mathcal{RO}^\dagger} + 4q^2_{\mathcal{TO}}}{2^\mu}, \frac{q_{\mathcal{TO}}}{2^{w-\log q_{\mathcal{RO}^\dagger}}}\right\} + \max\left\{\frac{4q^2_E + 4q^2_D}{2^\mu}, \frac{q_D}{2^{b-\log q_E}}, \frac{q_E}{2^{b-\log q_D}}\right\}.$$

$\mathcal{B}$ は $\mathcal{RO}$ へのクエリはなく, ν 回 RoS クエリを行い, そして, 実行時間は $(\mathcal{A}_1,\mathcal{A}_2)$ の時間プラス $\mathcal{O}(q_{\mathcal{RO}}+q_{\mathcal{RO}^*}+q_{\mathcal{RO}^\dagger}+q_{\mathcal{TO}}+q_E+q_D)$ である. $\mathsf{maxpk}_{\mathcal{AE}}$ は $\mathsf{maxpk}_{\mathcal{AE}} = \max_{\gamma\in\{0,1\}^*} \Pr[pk = \gamma : (pk,sk) \xleftarrow{\$} \mathsf{Gen}]$ と定義される公開鍵がコリジョンを起こす最大確率である. μ は mmr-source のミニエントロピーである. ♦

証明. ゲーム $\mathbf{G}_i$ での攻撃者 $\mathcal{A}$ のアドバンテージを $\mathsf{Adv}(\mathcal{A},\mathbf{G}_i)$ と書くことにする. ここで, ゲーム $\mathbf{G}_0$ を $\mathcal{WRO}$ モデルでの CDA ゲームとする. すなわち, $\mathsf{Adv}(\mathcal{A},\mathbf{G}_0) = \mathsf{Adv}^{\mathrm{cda}}_{\mathcal{AE},\mathcal{WRO}}(\mathcal{A}_1,\mathcal{A}_2)$ となる.

Game $\mathbf{G}_1$: ゲーム $\mathbf{G}_1$ では以下のイベントが起きた場合, $\mathcal{RO}_n$ は乱数を返信する.

- Bad_1 : チャレンジ暗号文を生成する際に $\mathcal{RO}$ にクエリされた値 M が, $\mathcal{A}_1$ によって $\mathcal{RO}_n$ にクエリされた場合.

そして, その他の手続きは $\mathbf{G}_0$ と同じとする. このとき, 以下の式が成り立つ.

補題 13 $|\mathsf{Adv}(\mathcal{A},\mathbf{G}_1) - \mathsf{Adv}(\mathcal{A},\mathbf{G}_0)| \le q_{\mathcal{RO}} \cdot \mathsf{maxpk}_{\mathcal{AE}}$.

証明. $\mathbf{G}_0$ と $\mathbf{G}_1$ の差は Bad_1 が起きた場合のみ生じる. よって, Difference Lemma [54] から, $|\mathsf{Adv}(\mathcal{B},\mathbf{G}_1) - \mathsf{Adv}(\mathcal{B},\mathbf{G}_0)| \le \Pr[\mathsf{Bad}_1]$ となる.

次に $\Pr[\mathsf{Bad}_1]$ を評価する. $\mathcal{A}_1$ は pk は手に入らず, **Enc** のクエリによって $\mathcal{RO}_n$ の入力に入っているため, pk を含むクエリを $\mathcal{RO}_n$ にするためには, 偶然 pk を選んで $\mathcal{RO}$ にクエリするしかない. $\mathcal{RO}$ へのクエリ回数は $q_{\mathcal{RO}}$ 回なので, $\Pr[\mathsf{Bad}_1] \le q_{\mathcal{RO}} \cdot \mathsf{maxpk}_{\mathcal{AE}}$ となる.

■

Game $\mathbf{G}_2$: このゲームでは, 暗号文 $\mathbf{c} \leftarrow \mathbf{Enc}^{\mathcal{RO}_n}(pk,\mathbf{m}_b;\mathbf{r})$ を IND-SIM のシミュレータ $\mathcal{S}^{\mathcal{RO}_n}(pk,\omega)$ に変更する. その他の手続きは $\mathbf{G}_1$ と同じとする.

そして以下の補助定理が成り立つ.

補題 14 $|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_2) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_1)| \leq \mathsf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S},\mathcal{RO}_n}(\mathcal{B})$.

証明. 以下, $|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_2) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_1)|$ が無視できる確率でさえられるならば, 任意のシミュレータ $\mathcal{S}$ に対して, $\mathcal{RO}$ モデルで $\mathcal{AE}$ の IND-SIM 安全性を破ることができる攻撃者 $\mathcal{B}$ を構成できることを示す. 図 4.8 はゲーム $\mathbf{G}_2$, $\mathcal{B}$ の構造, $\mathcal{B}$ による $\mathcal{WRO}$ モデルでのシミュレーション $\mathsf{SimB} = (\mathsf{SimB}_{\mathcal{RO}_n}, \mathsf{SimB}_{\mathcal{RO}^*_v}, \mathsf{SimB}_{\mathcal{RO}^\dagger_w}, \mathsf{SimB}_{\mathcal{TO}}, \mathsf{SimB}_E, \mathsf{SimB}_D)$ である. ここで, $\mathcal{B}$ は $\mathcal{RO}$ へクエリしないことに注意されたい. $\mathcal{B}$ は SimB を用いて $\mathcal{A}_1$ と $\mathcal{A}_2$ から $\mathcal{WRO}$ へのクエリレスポンスをシミュレートする. そして, $\mathbf{G}_1$ と $\mathbf{G}_2$ ではともに $\mathbf{Enc}$ による $\mathcal{RO}_n$ へのクエリが含まれている. よって, $\mathcal{WRO}$ はパーフェクトにシミュレートできているので, $\mathcal{A}$ は $\mathbf{G}_1$ と $\mathbf{G}_2$ を識別できない. そして, IND-SIM ゲームで $\beta = 1$ の場合, $\mathcal{A}$ のインターフェースは $\mathbf{G}_1$ と同じで, $\beta = 0$ の場合, $\mathcal{A}$ のインターフェースは $\mathbf{G}_2$ と同じになる.

よって, $|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_2) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_1)|$ が無視できない確率となる場合, $\mathcal{B}$ は $\mathcal{AE}$ の IND-SIM 安全性を破ることができる.

■

Game $\mathbf{G}_3$: このゲームでは, $\mathcal{A}_2$ が $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ に関するクエリを $\mathcal{RO}^*_v, \mathcal{RO}^\dagger_w, \mathcal{TO}$, または $\mathsf{IC}_{a,b} = (E, D)$ にしたときに, 出力を乱数とする. すなわち, このゲームでは $\mathcal{A}_1$ と $\mathcal{A}_2$ が $\mathcal{RO}^*_v, \mathcal{RO}^\dagger_w, \mathcal{TO}$, $\mathsf{IC}_{a,b} = (E, D)$ からは $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ を共有できない. そして, これ以外の手続きは $\mathbf{G}_2$ と同じとする.

補題 15

$$\begin{aligned}|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_3) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_2)| \leq & \frac{4q^2_{\mathcal{RO}^*} + 4q^2_{\mathcal{RO}^\dagger}}{2^\mu} + \max\left\{\frac{4q^2_{\mathcal{TO}}}{2^\mu}, \frac{4q^2_{\mathcal{TO}}}{2^w}\right\} \\ & + \max\left\{\frac{4q^2_E + 4q^2_D}{2^\mu}, \frac{4q^2_E + 4q^2_D}{2^b}\right\}.\end{aligned}$$

証明 $\mathbf{G}_2$ と $\mathbf{G}_3$ の差は, $\mathcal{A}_1$ と $\mathcal{A}_2$ が $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ に関する同じクエリを $\mathcal{RO}^*_v, \mathcal{RO}^\dagger_w, \mathcal{TO}$ または $\mathsf{IC}_{a,b} = (E, D)$ に行った場合に生じる. $\mathcal{A}_1$ と $\mathcal{A}_2$ が $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ に関する共通のクエリを $\mathcal{RO}^*_v$ に行うイベントを $\mathsf{Bad}_{\mathcal{RO}^*}$ と書くことにする. 同様に, イベント $\mathsf{Bad}_{\mathcal{RO}^\dagger}$, $\mathsf{Bad}_{\mathcal{TO}}$, Bad_E, Bad_D を定義する. そして, Difference Lemma [55] から以下の式が成り立つ.

$$\begin{aligned}|\mathsf{Adv}(\mathcal{B}, \mathbf{G}_3) - \mathsf{Adv}(\mathcal{B}, \mathbf{G}_2)| \leq & \Pr[\mathsf{Bad}_{\mathcal{RO}^*} \vee \mathsf{Bad}_{\mathcal{RO}^\dagger} \vee \mathsf{Bad}_{\mathcal{TO}} \vee \mathsf{Bad}_E \vee \mathsf{Bad}_D] \\ \leq & \Pr[\mathsf{Bad}_{\mathcal{RO}^*}] + \Pr[\mathsf{Bad}_{\mathcal{RO}^\dagger}] + \Pr[\mathsf{Bad}_{\mathcal{TO}}] \\ & + \Pr[\mathsf{Bad}_E] + \Pr[\mathsf{Bad}_D]\end{aligned}$$

Game $\mathbf{G}_2$

$\beta \xleftarrow{\$} \{0,1\}$

$(pk, sk) \xleftarrow{\$} \mathsf{Gen}$

$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow \mathcal{A}_1^{\mathcal{RO}_n, \mathcal{RO}_v^*, \mathcal{RO}_w^\dagger, \mathcal{TO}, \mathsf{IC}_{a,b}}$

$\mathbf{c} \leftarrow \mathbf{Enc}^{F.hon}(pk, \mathbf{m}_\beta; \mathbf{r})$

$\mathbf{c}' \leftarrow \mathcal{S}^{\mathcal{RO}_n}(pk, \omega)$

$\beta' \leftarrow \mathcal{A}_2^{\mathcal{RO}_n, \mathcal{RO}_v^*, \mathcal{RO}_w^\dagger, \mathcal{TO}, \mathsf{IC}_{a,b}}(pk, \mathbf{c}')$

return $(\beta = \beta')$

$\mathcal{B}^{\mathsf{RoS}}(pk)$

$\beta \xleftarrow{\$} \{0,1\}$

$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow \mathcal{A}_1^{\mathsf{SimB}}$

$\mathbf{c} \leftarrow \mathsf{RoS}(\mathbf{m}_\beta, \mathbf{r})$

$\beta' \leftarrow \mathcal{A}_2^{\mathsf{SimB}}(pk, \mathbf{c})$

If $\beta = \beta'$ then **return** 1

Otherwise **return** 0

$\mathsf{SimB}_E(k, x)$

If $\mathsf{E}[k, x] = \perp$,

$y \xleftarrow{\$} \{0,1\}^b \backslash T^+[k]$

$\mathsf{E}[k, x] \leftarrow y, \mathsf{D}[k, y] \leftarrow x,$

$T^+[k] \xleftarrow{\cup} \{y\}, T^-[k] \xleftarrow{\cup} \{x\}$

return $\mathsf{E}[k, x]$

$\mathsf{SimB}_{\mathcal{RO}_n}(M)$

If $\mathsf{F}[M] = \perp$, $\mathsf{F}[M] \xleftarrow{\$} \{0,1\}^n$

If $\mathsf{F}[M] \neq \perp$, and M is posed by **Enc**,

$\mathsf{F}[M] \xleftarrow{\$} \{0,1\}^n$

return $\mathsf{F}[M]$

$\mathsf{SimB}_{\mathcal{RO}_v^*}(M)$

If $\mathsf{F}^*[M] = \perp$, $\mathsf{F}^*[M] \xleftarrow{\$} \{0,1\}^v$

return $\mathsf{F}^*[M]$;

$\mathsf{SimB}_{\mathcal{RO}_w^\dagger}(M)$

If $\mathsf{F}^\dagger[M] = \perp$ then $\mathsf{F}^\dagger[M] \xleftarrow{\$} \{0,1\}^w$

return $\mathsf{F}^\dagger[M]$;

$\mathsf{SimB}_{\mathcal{TO}}(y)$

If $\exists_1 M$ s.t. $\mathsf{F}^\dagger[M] = y$ then **return** M

Otherwise **return** $\perp$

$\mathsf{SimB}_D(k, y)$

If $\mathsf{D}[k, y] = \perp$,

$x \xleftarrow{\$} \{0,1\}^b \backslash T^-[k];$

$\mathsf{E}[k, x] \leftarrow y, \mathsf{D}[k, y] \leftarrow x,$

$T^+[k] \xleftarrow{\cup} \{y\}, T^-[k] \xleftarrow{\cup} \{x\}$

return $\mathsf{D}[k, y]$

図.4.8　ゲーム $\mathbf{G}_2$ と攻撃者 $\mathcal{B}$ によるシミュレーション SimB

$\mathbf{G}_2$ と $\mathbf{G}_3$ では, $\mathcal{A}_1$ と $\mathcal{A}_2$ は $\mathbf{c}$ から $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$, $\mathcal{WRO}$ の入出力に関する情報は得られない. そして, $\mathcal{RO}_n$ へのクエリに対して, $\mathbf{G}_2$ と $\mathbf{G}_3$ の $\mathcal{A}_2$ のインターフェースは同じになる. よって, このようなクエリを行うためには, ミニエントロピー μ, $\mathcal{RO}_w^\dagger$ の出力, (E, D) の出力のいずれかを推定する必要がある. そして, 誕生日解析 (詳しくは付録を参照されたい) により, $\mathcal{RO}^*$ と $\mathcal{RO}^\dagger$ の出力の推定が当たる確率はそれぞれ $(2q_{\mathcal{RO}^*})^2/2^\mu$, $(2q_{\mathcal{RO}^\dagger})^2/2^\mu$ となる. $\mathcal{TO}$

に関して, $\mu < w$ の場合, 推定が当たる確率は $(2q_{\mathcal{TO}})^2/2^\mu$, それ以外は $(2q_{\mathcal{TO}})^2/2^w$ となる. E と D に関して, $\mu < b$ の場合, 推定が当たる確率はそれぞれ $(2q_E)^2/2^\mu$, $(2q_D)^2/2^\mu$, これ以外の場合, それぞれ $(2q_E)^2/2^b$, $(2q_D)^2/2^b$ となる.

以上より, 以下の式が成り立つ.

$$\begin{aligned}|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_3) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_2)| \leq & (4q^2_{\mathcal{RO}^*} + 4q^2_{\mathcal{RO}^\dagger})/2^\mu + \max\left\{4q^2_{\mathcal{TO}}/2^\mu, 4q^2_{\mathcal{TO}}/2^w\right\} \\ & + \max\left\{(4q_E^2 + 4q_D^2)/2^\mu, (4q_E^2 + 4q_D^2)/2^b\right\}\end{aligned}$$

■

Game $\mathbf{G}_4$: このゲームでは以下のイベントが起きた時に $\mathcal{RO}_n$ は乱数を返すようにする.

- Bad_2 : チャレンジ暗号文を生成するために **Enc** から $\mathcal{RO}_n$ へのクエリ M に対して, $\mathcal{A}_2$ が M を $\mathcal{RO}_n$ にクエリする.

他の手続きは $\mathbf{G}_3$ と同じである. そして, 以下の補助定理が成り立つ.

補題 16 $|\mathsf{Adv}(\mathcal{A}, \mathbf{G}_4) - \mathsf{Adv}(\mathcal{A}, \mathbf{G}_3)| \leq \frac{q_{\mathcal{RO}}}{2^\mu}$.

証明. まず, $\mathbf{G}_3$ と $\mathbf{G}_4$ に差が生じるならばならず Bad_2 が起こる. よって, Difference Lemma [55] より以下の式が成り立つ.

$$|\mathsf{Adv}(\mathcal{B}, \mathbf{G}_4) - \mathsf{Adv}(\mathcal{B}, \mathbf{G}_3)| \leq \Pr[\mathsf{Bad}_2]$$

以下, $\Pr[\mathsf{Bad}_2]$ を評価する. ここで, $\mathcal{RO}_n$ はランダムオラクルで, $\mathbf{c}$ は変更されて, $\mathcal{A}_1$ はサブオラクルに M に関するクエリを行わないため, $\mathcal{A}_2$ は r の情報に関してミニエントロピー μ 以上の情報を得ることはできない. よって, $\mathcal{RO}_n$ に M をクエリする方法は μ を推定するだけである. よって, $\Pr[\mathsf{Bad}_2] \leq \frac{q_{\mathcal{RO}}}{2^\mu}$ となる.

■

最後に, $\mathsf{Adv}(\mathcal{A}, \mathbf{G}_4)$ を評価する. ここで, $\mathbf{c}$ は $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ に関する情報は一切含んでおらず, $\mathcal{A}_2$ は $\mathcal{WRO}$ の出力が $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ とは独立に見えるため, $\mathbf{G}_4$ に勝つためには β をランダムに推定するしかない. よって, $\mathsf{Adv}(\mathcal{A}, \mathbf{G}_4) = 0$ となる.

以上より, 以下の式が成り立つ.

$$\begin{aligned}\mathsf{Adv}^{\mathrm{cda}}_{\mathcal{AE},\mathcal{WRO}}(\mathcal{A}_1,\mathcal{A}_2) \leq & \mathsf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S},\mathcal{RO}_n}(\mathcal{B}) + q_{\mathcal{RO}} \cdot \mathsf{maxpk}_{\mathcal{AE}} \\ & + (q_{RO} + 4q^2_{\mathcal{RO}^*} + 4q^2_{\mathcal{RO}^\dagger})/2^\mu + \max\left\{4q^2_{\mathcal{TO}}/2^\mu, 4q^2_{\mathcal{TO}}/2^w\right\} \\ & + \max\left\{(4q_E^2 + 4q_D^2)/2^\mu, (4q_E^2 + 4q_D^2)/2^b\right\}\end{aligned}$$

■

第 5 章

SHA-3 の最適な IFRO 安全性証明と高速化

5.1 本章の概要

本稿では, Sponge 構造の最適な IFRO 安全性の証明を与えるとともに, この証明の結果を用いて $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保するためのキャパシティサイズの範囲を示す.

Sponge [11] は Bertoni, Daemen, Peeters, Van Assche によって提案されたハッシュ関数の構造で, SHA-3 の構造に採用されている [13]. Sponge の利点は, Davies-Meyer 構造など従来の構造で用いられていた feed forward 演算が無いので, 内部で必要なメモリサイズを最少に抑えることができる点である. この点から, Sponge は軽量ハッシュ関数である Quark [2] や SPONGENT [17] の構造として採用されるなど, 省リソース環境に適したハッシュ関数設計に多く用いられている.

Sponge は図 5.1 のように, $r + c$ bit の置換 P を繰り返す構造である. Sponge 構造は, 任意長の値に対して, "absorbing step (吸収ステップ)" と呼ばれるステップで, 入力値を r-bit のブロックごとに分割し, 各ブロックを置換の入力の先頭 r bit に排他的論理和で吸収する. そして, "squeezing step (絞り出しステップ)" で置換の出力のうち r bit を出力値として置換を繰り返して r bit の整数倍の値を作る構造である. r bit の整数倍ではない長さの値を出力する場合は, 1 度 r bit の整数倍の値を生成して, そこから必要な長さを出力値とする. ここで, $r + c$ bit のうち c bit は Sponge の安全性に係わる値で "capacity" と呼ばれている. 具体的には, Bertoni らによって, P がランダム置換の場合に, Sponge が $\mathcal{O}(2^{c/2})$ 回のクエリを許しても IFRO 安全となることを証明した [12].

ここで, IFRO 安全性は Sponge を含め多くの構造に採用されている安全性で, ハッシュ関数

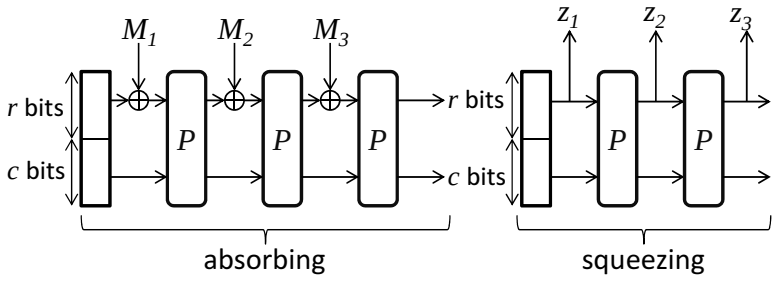

図.5.1　Sponge

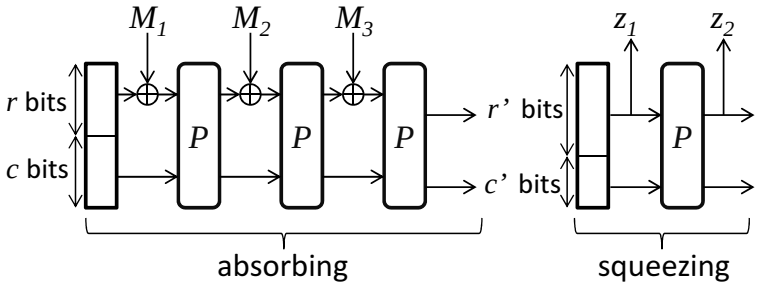

図.5.2　MSponge

の標準的な安全性の指標となっている [40]. IFRO 安全性は, $\mathcal{RO}$ モデルで安全な暗号アルゴリズムに $\mathcal{RO}$ の代わりに IFRO 安全なハッシュ関数を用いても同様の安全を持つことを保証する安全性である. 具体的には, 理想的な部品を用いたハッシュ関数 H に対してクエリ回数を q 回まで許した場合に IFRO 安全ならば, $\mathcal{RO}$ モデルで安全な任意の暗号システムに H を組み込んでも同様にクエリ回数が q 回までならば安全性が保証される [40, 51].

ここで, Sponge は P を固定したままキャパシティサイズ (c bit) を減らすと r bit は増えるため, 入力値 1 ブロック分と出力値 1 ブロック分の bit 長が長くなり, P の繰り返し回数を減らすことができるので, Sponge を高速化することができる. 一方で, Sponge は $\mathcal{O}(2^{c/2})$ 安全性の IFRO 安全性が証明されているが, P を固定したまま c を減らすと安全性が低下するといった問題が生じる.

ところで, absorbing step のキャパシティのコリジョンを用いると Sponge の IFRO 安全性を破ることができる. キャパシティのコリジョンは $\mathcal{O}(2^{c/2})$ 回のクエリで見つけることができるため, 結果として $\mathcal{O}(2^{c/2})$ 回のクエリで Sponge の IFRO 安全性が破れる. よって, Sponge の IFRO 安全性とこの攻撃のクエリ回数は一致するので, absorbing step のキャパシティサイズを減らすと安全性の低下に直結する.

一方で, squeezing step のキャパシティサイズを利用して $\mathcal{O}(2^{c/2})$ のクエリ回数で破る攻撃法は知られていない. また, absorbing step の 1 ブロック目のキャパシティの値は初期値のため固定値である. この固定値は攻撃者がコントロールできない値のため, キャパシティのコリジョンを用いた攻撃は適用できない. よって, 以下の 2 点が Sponge に関する研究課題として考えられる. 以降では, absorbing step の 1 ブロック目以外のキャパシティサイズ c を固定し, 1 ブロック目のキャパシティサイズを c_1, squeezing step のキャパシティサイズを c' として議論する. 特に, 高速化をねらいとして $c_1 \leq c, c' \leq c$ の場合に注目する.

1. Sponge が $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保できる c' の値 c'_0 を特定すること. さらに, c'_0 が $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保するために最適な値であることを示すこと. すなわち, $c' = c'_0$ で $\mathcal{O}(2^{c/2})$ の IFRO 安全性を破る攻撃法が存在することを示すこと.
2. Sponge が $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保できる c_1 の値 c'_1 を特定すること. さらに, c'_1 が $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保するために最適な値であることを示すこと. すなわち, $c_1 = c'_1$ で $\mathcal{O}(2^{c/2})$ の IFRO 安全性を破る攻撃法が存在することを示すこと.

上記の 2 点が示せれば, Sponge の $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保できる最適なキャパシティサイズを選ぶことができる. そして, 最適なキャパシティサイズを選ぶことで, P の呼び出し回数が減らすことができる. SHA-3 は Sponge を採用したハッシュ関数なので, SHA-3 にこの結果を適用できる. よって, この結果により, SHA-3 を高速化することができる. また, その他の Sponge を用いたハッシュ関数も同様に高速化できる.

5.1.1 本研究成果

上記の課題に対して, まず, c'_0 の値を特定する. 以下の議論では, $c' \leq c$ とした Sponge を MSponge と呼ぶことにする. 5.2 節で MSponge が $\mathcal{O}\left(\min\{q_{\mathsf{mcoll}}(d, c^*), 2^{c'}/d, 2^{c/2}\}\right)$ のクエリ回数で IFRO 安全性が破られないことを証明する. ここで, $c^* = c - c'$ で, $q_{\mathsf{mcoll}}(d, c^*)$ は MSponge の出力の c^* bit で d 個のマルチコリジョンを見つけるためのクエリ回数を表す. d は証明者が自由に設定できるパラメータである. d が増えると $2^{c'}/d$ は小さくなるが, $q_{\mathsf{mcoll}}(d, c^*)$ は大きくなるため, d として $q_{\mathsf{mcoll}}(d, c^*) = 2^{c'}/d$ となるパラメータを設定することで, 上記のクエリ回数が最大となる. 以下, $q_{\mathsf{mcoll}}(d, c^*) = 2^{c'}/d$ となる d を d^* と書くことにする. この表記を用いると, クエリ回数は $\mathcal{O}\left(\min\{q_{\mathsf{mcoll}}(d^*, c^*), 2^{c/2}\}\right)$ となる.

次に, 5.3 節で, squeezing step のキャパシティサイズを利用した $q_{\mathsf{mcoll}}(d^*, c^*)$ のクエリ回数で IFRO 安全性を破る攻撃法を示す. そして, absorbing step のキャパシティのコリジョンを利用した $\mathcal{O}(2^{c/2})$ のクエリ回数で IFRO 安全性を破る攻撃法と合わせると, IFRO 安全性を

Hash Function	Capacity	Absorb	Squeeze	Security
Sponge (Original) [33]	$c = c_1 = c' = 256$	9	7	2^{128}
MSponge (Optimalized)	$c = c_1 = 256, , c' = 133$	9	1	2^{128}
MSponge*	$c = 256, c_1 = 128, c' = 133$	5	1	2^{128}

表 5.1.1. この表は PHOTON-256/32/32 のパラメータを用いた時の Sponge, MSponge, MSponge* の P の呼び出し回数を比較したものである．Sponge, MSponge, MSponge* それぞれで用いる P は同じ性能であると仮定する．

破るクエリ回数は $\mathcal{O}\left(\min\{q_{\mathsf{mcoll}}(d^*, c^*), 2^{c/2}\}\right)$ で, 上記の IFRO 安全性が最適であることが言える．

次に, 5.4 章で, クエリ回数 $\mathcal{O}\left(\min\{q_{\mathsf{mcoll}}(d^*, c^*), 2^{c/2}\}\right)$ をもとに, $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保するための c', d^* の値を調べて, $c' = c/2 + \log_2 c$ かつ $d^* = c/2 - \log_2 c$ で $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保することができることを示す．Sponge の仕様は $c' = c$ であったが, 本結果により, MSponge で $c' = c/2 + \log_2 c$ $(< c)$ としても Sponge と同等の IFRO 安全性を担保することができる．

次に, c_1' の値を特定する．以降, $c_1 \leq c, c' \leq c$ とした Sponge を MSponge* と呼ぶことことにする．そして, 5.5 節で, MSponge* は, $c_1 \geq c/2$ であれば, $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保できることを示す．そして, absorbing ステップの第 1 ブロックの c_1 のサイズを利用した場合, $\mathcal{O}(2^{c_1/2})$ のクエリ回数で IFRO 安全性を破る攻撃法を示す．以上より, $c_1 = c/2$ が $\mathcal{O}(2^{c/2})$ の IFRO 安全性を担保できる最適な値となる．

最後に, 表 5.1.1 で Sponge, MSponge, MSponge* の速度を比較する．ここでは, Sponge を用いたハッシュ関数である PHOTON-256/32/32 のパラメータを用いる．PHOTON のパラメータは $r + c = 288$ bit で $c = 256$ である．このパラメータに対する Sponge の IFRO 安全性は 2^{128} なので, MSponge, MSponge* も 2^{128} の IFRO 安全性を持つようにパラメータを設定する．そして, 入力長を 256 bit, 出力長を 256 bit とした場合の, absorbing step と squeezing step での P の呼び出し回数を比較する．P の呼び出し回数は, Sponge が 16 回, MSponge が 10 回, MSponge* が 6 回となり, Sponge と比べて MSponge は 1.6 倍高速化できて, MSponge* は 2.6 倍高速化できていることになる．

Algorithm $\mathsf{MSponge}^P(n, m)$

1. parse $\mathsf{pad}(m)$ into r-bit blocks $(m_1, \ldots, m_\ell)$
2. $s = IV$
3. for $i = 1, \ldots, \ell$ do $s \leftarrow P(s \oplus (m_i \| 0^c))$
4. $w \leftarrow s$; $w_1 \leftarrow s$
5. for $j = 1, \ldots, \lceil n/r' \rceil$ do $w_{j+1} \leftarrow P(w_j)$; $w \leftarrow w || w_{j+1}[1, r']$
6. **return** $w[1, n]$

図.5.3　MSponge

5.1.2　関連研究

Guo, Peyrin, Poschmann が提案した PHOTON [33] は上記で議論してきた MSponge の構造を採用したハッシュ関数である．彼らは, IFRO 安全性については一切触れておらず, IFRO 安全性より弱い安全性である衝突困難性, 第 2 原像計算困難性, 一方向性を満たすための MSponge の最適なキャパシティサイズの値のみを与えている．本研究では, これらの 3 つの安全性概念を包含する IFRO 安全性について, Sponge の最適なキャパシティサイズを与える．

5.2　MSponge の IFRO 安全性証明の改良

本節では $\mathsf{MSponge}^P$ の IFRO 安全性の証明を与える．

5.2.1　MSponge 構造.

まず, MSponge 構造を説明する．P を t bit の置換とする．P^{-1} を P の逆演算とする．MSponge の部品に置換 P を用いたハッシュ関数を $\mathsf{MSponge}^P$ と書き, 内部の構造を図 5.3 で定義する．$\mathsf{MSponge}^P$ は任意長の値 m と 出力長 n を入力として, n bit の値を出力する．$\mathsf{pad} : \{0,1\}^* \to (\{0,1\}^r)^*$ を全単射関数とする．ただし, $\forall m \in \{0,1\}^*$ に対して, $\mathsf{pad}(m)$ の最後の r bit は 0^r 以外とする．IV を t bit の固定値とする．r を入力の 1 ブロックの長さ, r' を出力の 1 ブロックの長さ, $c := t - r$, $c' := t - r'$ とする．そして, ステップ 3 の s_i の c bit 成分 $s_i[r+1, t]$ を Absorbing ステップのキャパシティ, ステップ 4 の w_j の c' bit 成分 $w_j[r'+1, t]$ を Squeezing ステップのキャパシティと呼ぶ．また, Absorbing ステップと Squeezing ステッ

プどちらのステップのどちらを対象に議論していることが明らかな場合は, 単純にキャパシティと呼ぶことがある. ここで, $IV_1 := IV[1, r]$, $IV_2 := IV[r+1, t]$ とする.

5.2.2 $\mathsf{MSponge}^P$ の IFRO 安全性の証明.

次に, P をランダム置換として, $\mathsf{MSponge}^P$ の IFRO 安全性を証明する. ここで, P の逆演算を行うオラクルを P^{-1} と書く. そして, c^*bit の Q 個の乱数のうち d 個以上の値が同じ値となる確率を $p_{\mathsf{mcoll}}(Q, d, c^*)$ と書くことにする.

定理 5.1 $c' \leq c$ とする. シミュレータ $S = (S_F, S_I)$ が存在して, 任意の識別者 $\mathcal{D}$ に対して以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MSponge}^P, \mathcal{RO}, S}(\mathcal{A}) \leq \max\left\{ \frac{(d-1)q}{2^{c'}} + \frac{q(q-1)}{2^c} + Np_{\mathsf{mcoll}}(q_L, d, c^*), \frac{\sigma^2}{2^{c-1}} \right\} + \frac{\sigma(\sigma-1)}{2^{t+1}}.$$

ここで, $\mathcal{D}$ は L, R_F, R_I にそれぞれ q_L, q_F, q_I 回クエリできるとする. Real World では $(L, R_F, R_I) = (\mathsf{MSponge}^P, P, P^{-1})$, Ideal World では $(L, R_F, R_I) = (\mathcal{RO}, S_F, S_I)$ である. N を L へのクエリの入力ブロック数と出力ブロック数の和の最大値とする. d はフリーパラメータで上記のバウンドが最小となるように設定する. $\sigma := Nq_L + q$ とする. そして, これらのパラメータに対して, S のクエリ回数は最大 q で, S の動作時間は $\mathcal{O}(q)$ となる. ♦

5.2.3 証明の概要

本章では, 定理 5.2 の証明の概要を説明する.

$\mathsf{MSponge}^P$ が IFRO 安全であるとは, 任意の識別者 $\mathcal{D}$ が 3 つのオラクル (L, R_F, R_I) と対話し, Real World と Ideal World を識別できないようなシミュレータ $S = (S_F, S_I)$ を構成できることである.

この概要では, 次の例を用いて説明する: $\mathsf{MSponge}^P$ のパディングは省略する. また, L へのクエリ長を 1 ブロック (r bit) または 2 ブロック ($2r$ bit) の 2 パタンとして, L の出力長を 1 ブロック (r' bit) とする.

以下, 次の順番で証明の概要を説明する.

1. $\mathcal{D}$ が Real World と Ideal World を区別できないようにするために, S が対応すべきケースを列挙する.

2. それぞれのケースで S がシミュレートに失敗する確率を評価する.

5.2.3.1 S が対応すべきケース

まず, $\mathcal{D}$ が Real World $((L, R_F, R_I) = (\mathsf{MSponge}^P, P, P^{-1}))$ と Ideal World $((L, R_F, R_I) = (\mathcal{RO}, S_F, S_I))$ を識別できないための $\mathcal{D}$ が対話する対象は, L オラクルと R オラクルの 2 つに分類することができるので, $S = (S_F, S_I)$ が対応すべきケースは次の 2 つである.

- ケース 1 (R オラクルに対応するケース): Real World では $(R_F, R_I) = (P, P^{-1})$ なので, $S = (S_F, S_I)$ は (P, P^{-1}) をシミュレートする.
- ケース 2 (L オラクルに対応するケース): Real World は $(L, R_F, R_I) = (\mathsf{MSponge}^P, P, P^{-1})$ で, $\mathsf{MSponge}^P$ は L へのクエリに対して P にアクセスしてレスポンスを計算しているため, $\mathsf{MSponge}^P$ のある入出力と (P, P^{-1}) のある入出力に関係がある. S はこの関係をシミュレートしなければならない.

ケース 2 では, $\mathsf{MSponge}^P$ の入力長は 1 ブロックと 2 ブロックの 2 パタンを考えているので, 1 ブロックのケースと 2 ブロックのケースに分ける.

まず, この説明で使う記号を導入する. $\mathcal{D}$ のクエリ終了後の R_F へのクエリレスポンスリストを T_F, R_I へのクエリレスポンスリストを T_I とする. $T := T_F \cup T_I$ とする. 次に T に対する MSponge パスを定義する: $(x_1, y_1) \in T$ s.t. $x_1[r+1, t] = IV_2 \wedge m_1 = IV_1 \oplus x_1[1, r]$: $IV \xrightarrow{m_1} (x_1, y_1)$. また, $(x_1, y_1), (x_2, y_2) \in T$ s.t. $x_1[r+1, t] = IV_2 \wedge x_2[r+1, t] = y_1[r+1, t] \wedge m_1 = IV_1 \oplus x_1[1, r] \wedge m_2 = y_1[1, r] \oplus x_2[1, r]$: $IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} (x_2, y_2)$ または $IV \xrightarrow{m_1 \| m_2} (x_2, y_2)$ と表現する. ここで, $m_1 = IV_1 \oplus x_1[1, r]$, $m_2 = y_1[1, r] \oplus x_2[1, r]$ とする. そして, これらを MSponge パスと呼ぶことにする.

以下, G_T を MSponge パスの集合とする. また, 説明を簡単にするために, 順番とクエリの種類を表す記号を導入する. まず, クエリの順番について, (x_i, y_i) の右下に順番を記載する. 例えば, (x_i, y_i) が先に決まって, 次に (x_j, y_j) が決まる場合, $(x_i, y_i)_1, (x_j, y_j)_2$ と書くことにする. また, $(x_i, y_i) \in T_F$ の場合, $\overrightarrow{(x_i, y_i)}$ と書き, $(x_i, y_i) \in T_I$ の場合, $\overleftarrow{(x_i, y_i)}$ と書くことにする.

それぞれのケースで S が具体的に実現すべきことは, 以下の通りである.

- ケース 2-1 (1 ブロックケース): Real World では, 以下が成り立つ.

$$\forall \left(IV \xrightarrow{m_1} (x_1, y_1) \right) \in G_T : y_1[1, r'] = L(m_1).$$

よって, Ideal World でも同じ関係が成り立つように S を構成しなければならない.

- ケース 2-2 (2 ブロックケース): Real World では, 以下が成り立つ.

$$\forall \left(IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} (x_2, y_2) \right) \in G_T : y_2[1, r'] = L(m_1 \| m_2).$$

よって, Ideal World でも同じ関係が成り立つように S を構成しなければならない.

上記のケース 1, ケース 2-1, ケース 2-2 の 3 ケースに対応できる S を構成できれば, $\mathcal{D}$ は Real World と Ideal World を識別できない. よって, $\mathcal{D}$ が識別できる確率 $p_{\mathcal{D}}$ は以下の通りである.

$$\begin{aligned} p_{\mathcal{D}} \leq & \Pr[\text{ケース 1 のシミュレートに失敗}] \\ & + \Pr[\text{ケース 2-1 のシミュレートに失敗}] \\ & + \Pr[\text{ケース 2-2 のシミュレートに失敗}] \end{aligned} \tag{5.1}$$

以下, $\mathcal{D}$ の L へのクエリ回数を q_L, R_F へのクエリ回数を q_F, R_I へのクエリ回数を q_I として, 上記の確率を評価する.

5.2.3.2 ケース 1 でのシミュレート方法と失敗確率

ケース 1 に対応する S の振る舞いとして, 初出のクエリに対する出力がランダムになるように S_F, S_I を定義する.

この場合, PRF/PRP switching lemma [10] (詳しくは付録を参照されたい) より, ケース 1 のシミュレートに失敗する確率は以下の通りである.

$$\Pr[\text{ケース 1 のシミュレートに失敗}] \leq \frac{(q_F + q_I)^2}{2^{t+1}} \tag{5.2}$$

5.2.3.3 ケース 2-1 でのシミュレート方法と失敗確率

$\forall \left(IV \xrightarrow{m_1} (x_1, y_1) \right) \in G_T$ を考えて, 以下の場合分けを行う.

- ケース a: $\overrightarrow{(x_1, y_1)}$.
- ケース b: $\overleftarrow{(x_1, y_1)}$.

よって,

$$\begin{aligned} \Pr[\text{ケース 2-1 のシミュレートに失敗}] \leq & \Pr[\text{ケース a でシミュレートに失敗}] \\ & + \Pr[\text{ケース b のシミュレートに失敗}] \end{aligned}$$

ケース a. ケース a に対応する S_F の動作を以下のように定義する. クエリ x_1 に対して, 以下の方法でレスポンス y_1 を定義する.

1. $m_1 \leftarrow x_1[1,r] \oplus IV_2$
2. $y_1[1,r'] \leftarrow \mathcal{RO}(m_1)$
3. $y_1[r'+1,t] \xleftarrow{\$} \{0,1\}^{c'}$

この手続きにより, 確率 1 で $y_1[1,r'] = \mathcal{RO}(m_1)$ となる. すなわち,

$$\Pr[\text{ケース a でシミュレートに失敗}] = 0$$

ここで, ケース 2-2 の議論で $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1,y_1)}\right) \in G_T$ となる $\overrightarrow{(x_1,y_1)}$ が出てきた場合, 上記の手続きを用いることを覚えておいていただきたい.

ケース b. ケース b について, 以下の関係が成り立つ.

$$\Pr[\text{ケース b でシミュレートに失敗}] \leq \Pr[\text{ケース b が発生}]$$

ここで, $IV \xrightarrow{m_1} \overleftarrow{(x_1,y_1)}$ なので, $x_1[r+1,r+c] = IV_2$ となる. そして, x_1 はランダムに選ばれることから, $x_1[r+1,r+c] = IV_2$ となる確率は $1/2^c$ となる. $\mathcal{D}$ は R_I に q_I 回クエリするため,

$$\Pr[\text{ケース b が発生}] \leq \frac{q_I}{2^c}$$

ケース 2-1 の失敗確率. 以上より,

$$\Pr[\text{ケース 2-1 のシミュレートに失敗}] \leq \frac{q_I}{2^c}. \tag{5.3}$$

5.2.3.4 ケース 2-2 でのシミュレート方法と失敗確率

$\forall \left(IV \xrightarrow{m_1} (x_1,y_1) \xrightarrow{m_2} (x_2,y_2)\right) \in G_T$ を考えて, 以下の場合分けを行う.

- ケース a: $\overrightarrow{(x_1,y_1)}$.
  - ケース a1: (x_1,y_1) の候補が 1 つ.
    - ケース a11: $\overrightarrow{(x_2,y_2)}$ かつ $y_1[r+1,c] \neq IV_2$ となる.
      - ケース a111: $(x_1,y_1)_1, (x_2,y_2)_2$.

・ケース a112: $(x_1, y_1)_2, (x_2, y_2)_1$.

∗ ケース a12: $\overrightarrow{(x_2, y_2)}$ かつ $y_1[r+1, c] = IV_2$ となる.

∗ ケース a13: $\overleftarrow{(x_2, y_2)}$

・ケース a131: $(x_1, y_1)_1, (x_2, y_2)_2$.

・ケース a132: $(x_1, y_1)_2, (x_2, y_2)_1$.

– ケース a2: (x_1, y_1) の候補が複数存在する. すなわち,

$$\exists \left(IV \xrightarrow{m_1^*} (x_1^*, y_1^*) \xrightarrow{m_2^*} (x_2, y_2) \right) \in G_T \text{ s.t. } (x_1^*, y_1^*) \neq (x_1, y_1)$$

- ケース b: $\overleftarrow{(x_1, y_1)}$ によって定義される.

以上の場合分けから, 以下の式が成り立つ.

$$\begin{aligned}
&\Pr[\text{ケース 2-2 でシミュレートに失敗}] \\
&\quad \leq \Pr[\text{ケース a111 でシミュレートに失敗}] + \Pr[\text{ケース a112 でシミュレートに失敗}] \\
&\qquad + \Pr[\text{ケース a12 でシミュレートに失敗}] + \Pr[\text{ケース a131 でシミュレートに失敗}] \\
&\qquad + \Pr[\text{ケース a132 でシミュレートに失敗}] + \Pr[\text{ケース a2 でシミュレートに失敗}] \\
&\qquad + \Pr[\text{ケース b でシミュレートに失敗}]
\end{aligned}$$

ケース a111. ケース a111 は $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}_2 \right) \in G_T$ かつ $x_2[r+1, t] \neq IV_2$ である. ケース a111 に対応する S_F の動作を以下のように定義する.

まず, $IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1$ に対応する S_F へのクエリ x_1 に対するレスポンス y_1 をケース 2-1 のケース a のシミュレート方法を用いて定義する.

次に, $\overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}_2$ に対応する S_F へのクエリ x_2 に対するレスポンス y_2 を以下のように定義する.

1. $y_1[r+1, t] = x_2[r+1, t]$ となることを手掛かりに, $IV \xrightarrow{m_1} (x_1, y_1)$ を見つける
2. $m_2 \leftarrow x_2[1, r] \oplus y_1[1, r]$
3. $y_2[1, r'] \leftarrow \mathcal{RO}(m_1 \| m_2)$
4. $y_2[r'+1, t] \xleftarrow{\$} \{0,1\}^{c'}$

$y_1[r+1, c] \neq IV_2$ なので, クエリ x_2 に対して, ケース 2-1 のケース a とこのケースが混同することはなく, 上記のステップを実行できる. そして, 上記の S_F の振る舞いから確率 1 で $y_2[1, r'] = \mathcal{RO}(m_1 \| m_2)$ となる. すなわち,

$$\Pr[\text{ケース a111 のシミュレートに失敗}] = 0.$$

ケース a112. ケース a112 は $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_2 \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}_1\right) \in G_T$ かつ $x_2[r+1, t] \neq IV_2$ である. ケース a112 について, 以下の関係が成り立つ.

$$\Pr[\text{ケース a112 のシミュレートに失敗}] \leq \Pr[\text{ケース a112 が発生}]$$

以下, ケース a112 が発生する確率を評価する.

ここで, $\mathcal{RO}$ の出力の後半 c^* bit に d-multi-collision が発生するイベントを $\mathbf{MColl}_d$ と書くことにする. d-multi-collision イベントは, d 個のメッセージ $v_1, \ldots, v_d$ が存在し, $\mathcal{RO}(v_1)[r+1, r'] = \mathcal{RO}(v_2)[r+1, r'] = \cdots = \mathcal{RO}(v_d)[r+1, r']$ となることである. $\mathbf{MColl}_d$ を用いると以下の式が成り立つ.

$$\begin{aligned}
&\Pr[\text{ケース a112 のシミュレートに失敗}] \\
&\quad \leq \Pr[\text{ケース a112 が発生}] \\
&\quad = \Pr[\text{ケース a112 が発生} \wedge \neg\mathbf{MColl}_d] + \Pr[\text{ケース a112 が発生} \wedge \mathbf{MColl}_d] \\
&\quad \leq \Pr[\text{ケース a112 が発生} \mid \neg\mathbf{MColl}_d] + \Pr[\mathbf{MColl}_d]
\end{aligned}$$

まず, $\Pr[\text{ケース a112 が発生} \mid \neg\mathbf{MColl}_d]$ すなわち, $\neg\mathbf{MColl}_d$ の場合に, $\overrightarrow{(x_1, y_1)}_2 \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}_1$ となる確率を評価する. ケース 2-1 の手続きから, $y_1[1, r'] = \mathcal{RO}(m_1)$ となるため, $x_2[r+1, r'] := \mathcal{RO}(m_1)[r+1, r']$ とすることで, 確率 1 で $y_1[r+1, r'] = x_2[r+1, r']$ となる. 残りの c' bit について, $y_1[r'+1, t]$ は $x_2[r'+1, t]$ とは独立にランダムに選ばれるため, $\overrightarrow{(x_1, y_1)}_2 \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}_1$ となる確率は $1/2^{c'}$ となる.

そして, $y_1[r+1, r'] = x_2[r+1, r'] = \mathcal{RO}(m_1)[r+1, r']$ に注意すると, $\neg\mathbf{MColl}_d$ から, $(x_2, y_2) \in T_F$ を 1 つ固定したときの $y_1[r+1, t] = x_2[r+1, t]$ となる $(x_1, y_1) \in T_F$ の個数は高々 $d-1$ 個である.

そして, T_F の要素は高々 q_F 個存在するので, 以下の式が成り立つ.

$$\Pr[\text{ケース a112 が発生} \mid \neg\mathbf{MColl}_d] \leq \frac{(d-1)q_F}{2^{c'}}.$$

また, この節では $\Pr[\mathbf{MColl}_d]$ の値は議論せずそのままにしておく. この値は 5.3 節で議論する.

ケース a12. ケース a12 は $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)} \xrightarrow{m_2} \overrightarrow{(x_2, y_2)}\right) \in G_T$ かつ $x_2[r+1, t] = IV_2$ である. ケース a12 について, 以下の関係が成り立つ.

$$\Pr[\text{ケース a12 のシミュレートに失敗}] \leq \Pr[\text{ケース a12 が発生}]$$

以下, ケース a12 が起こる確率, すなわち, $y_1[r+1,t] = x_2[r+1,t] = IV_2$ となる確率は, y_1 はランダムに選ばれることから, $1/2^c$ となる.

そして, T_F には高々 q_F 個の要素が存在するため, 以下の式が成り立つ.

$$\Pr[\text{ケース a12 が発生}] \leq \frac{q_F}{2^c}.$$

ケース a131. ケース a131 は $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \overleftarrow{(x_2, y_2)}_2\right) \in G_T$ である. ケース a131 について, 以下の関係が成り立つ.

$$\Pr[\text{ケース a131 のシミュレートに失敗}] \leq \Pr[\text{ケース a131 が発生}]$$

以下, ケース a131 が起こる確率, すなわち, $\overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \overleftarrow{(x_2, y_2)}_2$ となる確率は, x_2 はランダムに選ばれることから, $1/2^c$ である.

そして, T_F には高々 q_F 個の要素が, T_I には高々 q_I 個の要素が存在するため, 以下の式が成り立つ.

$$\Pr[\text{ケース a131 が発生}] \leq \frac{q_F q_I}{2^c}.$$

ケース a132. ケース a132 は $\left(IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_2 \xrightarrow{m_2} \overleftarrow{(x_2, y_2)}_1\right) \in G_T$ である. ケース a132 について, 以下の関係が成り立つ.

$$\Pr[\text{ケース a132 のシミュレートに失敗}] \leq \Pr[\text{ケース a132 が発生}]$$

以下, ケース a132 が起こる確率, すなわち $\overrightarrow{(x_1, y_1)}_2 \xrightarrow{m_2} \overleftarrow{(x_2, y_2)}_1$ が起こる確率は, y_1 は x_2 と独立にランダムに選ばれることから, $1/2^c$ となる.

そして, T_F には高々 q_F 個の要素が, T_I には高々 q_I 個の要素が存在するため, 以下の式が成り立つ.

$$\Pr[\text{ケース a132 が発生}] \leq \frac{q_F q_I}{2^c}.$$

ケース a2. ケース a2 について, 以下の関係が成り立つ.

$$\Pr[\text{ケース a2 のシミュレートに失敗}] \leq \Pr[\text{ケース a2 が発生}]$$

ケース a2 では, $y_1^{(1)}[r+1,t] = y_1^{(2)}[r+1,t] = x_2[r+1,t]$ となるペア $(x_1^{(1)}, y_1^{(1)}), (x_1^{(2)}, y_1^{(2)})$ が存在する. S_F の出力はランダムに選ばれることから, 誕生日解析 (詳しくは付録を参照され

たい) を適用すると以下の式が成り立つ.

$$\Pr[\text{ケース a2 が発生}] \leq \frac{q_F^2}{2^{c+1}}.$$

ケース b. ケース b について, 以下の関係が成り立つ.

$$\Pr[\text{ケース b のシミュレートに失敗}] \leq \Pr[\text{ケース b が発生}]$$

x はランダムに選ばれることから, $x[r+1,t] = IV_2$ となる $(x_1, y_1) \in T_I$ が存在する確率は $q_I/2^c$ なので, 以下の式が成り立つ.

$$\Pr[\text{ケース b が発生}] \leq \frac{q_I}{2^c}.$$

ケース 2-2 の失敗確率以上より, 以下の式が成り立つ.

$$\begin{aligned}
&\Pr[\text{ケース 2-2 でシミュレートに失敗}] \\
&\quad \leq \frac{(d-1)q_F}{2^{c'}} + \Pr[\mathbf{MColl}_d] + \frac{q_F}{2^c} + \frac{q_F q_I}{2^c} + \frac{q_F q_I}{2^c} + \frac{q_F^2}{2^{c+1}} + \frac{q_I}{2^c} \\
&\quad \leq \frac{dq_F}{2^{c'}} + \Pr[\mathbf{MColl}_d] + \frac{q_F^2 + 4q_F q_I + 2(q_F + q_I)}{2^{c+1}} \qquad (5.4)
\end{aligned}$$

5.2.3.5 識別するための計算量

以上より, $\sigma = q_L + q_F + q_I$ とすると, 式 (5.1), (5.2), (5.3) (5.4) より以下の式が成り立つ.

$$p_{\mathcal{D}} \leq \underbrace{\underbrace{\frac{(q_F+q_I)^2}{2^{t+1}}}_{\text{式 (5.2)}} + \underbrace{\frac{q_I}{2^c}}_{\text{式 (5.3)}} + \underbrace{\frac{dq_F}{2^{c'}} + \Pr[\mathbf{MColl}_d] + \frac{q_F^2 + 4q_F q_I + 2(q_F+q_I)}{2^{c+1}}}_{\text{式 (5.4)}}}_{\text{式 (5.1)}}$$

ここで, d は証明者が設定できる free パラメータである. そして, d が大きくなると, $\Pr[\mathbf{MColl}_d]$ の値は小さくなる. 一方で, d が大きくなると, $\frac{dq_F}{2^{c'}}$ の値も大きくなる. そこで, $dq_F/2^{c'} = \Pr[\mathbf{MColl}_d]$ となる d を d^* と設定すると, $p_{\mathcal{D}}$ の上界値が最も小さくなり, 以下の式が成り立つ.

$$p_{\mathcal{D}} = \mathcal{O}\left(\frac{\sigma^2}{2^t} + \frac{\sigma^2}{2^c} + \frac{d^*\sigma}{2^{c'}}\right)$$

$t \geq c$ なので, 以下の式が成り立つ.

$$p_{\mathcal{D}} = \mathcal{O}\left(\frac{\sigma^2}{2^c} + \frac{d^*\sigma}{2^{c'}}\right)$$

以上より, $\mathcal{D}$ のクエリ回数を $2^{c/2}$ または $2^{c'}/d^*$ まで許すと, $p_{\mathcal{D}}$ は定数となる. よって, $\sigma = \mathcal{O}(\min\{2^{c/2}, 2^{c'}/d^*\})$ とすると, $p_{\mathcal{D}}$ は定数以下にに抑えることができる.

5.2.4 S の構成

(L, R_F, R_I) のクエリーレスポンスの定義

(L, R_F, R_I) と対話する任意の $\mathcal{D}$ に対して, Real World $(L, R_F, R_I) = (\mathsf{MSponge}^P, P, P^{-1})$ と Ideal World $(L, R_F, R_I) = (\mathcal{RO}, S_F, S_I)$ を識別できないようなシミュレータ (S_F, S_I) を構成する.

ここでは, 仕様通りの MSponge を考えるため, (L, R_F, R_I) のクエリーレスポンスも MSponge の仕様と合わせる. L の入力はメッセージ m に加えて, L の出力長 n も入力パラメータとなる. Real World では $L(n, m) = \mathsf{MSponge}^P(n, m)$, Ideal World では入出力長が任意長となる $\mathcal{RO}$ を扱い, $L(n, m) = \mathcal{RO}(m)[1, n]$ とする. また, R_F と R_I の入出力長は t bit である.

記号の定義

まず, S を構成する前に, いくつか記号を定義する.

定義 24 (R_F と R_I の入出力を記録するリスト)

- T_F を R_F のクエリーレスポンスを記録するリストとする.
- T_I を R_I のクエリーレスポンスを記録するリストとする.
- $T = T_F \cup T_I$ とする.

次に MSponge パスについて, ここでは仕様通りの MSponge 構造を考えるため, MSponge パスも MSponge の仕様に対応することとする.

定義 25 (MSponge ブロック) $(x_1, y_1), (x_2, y_2) \in T \wedge y_1[r+1, t] = x_2[r+1, t] \wedge m_2 = y_1[1, r] \oplus x_2[1, r] \Leftrightarrow (x_1, y_1) \xrightarrow{m_2} (x_2, y_2)$ として, これを MSponge ブロックと呼ぶことにする. T_b を MSponge ブロックの集合とする.

定義 26 (初期ブロックとダミーブロック) $(x_1, y_1) \in T \wedge x_1[r+1, t] = IV_2 \wedge m_1 = x_1[1, r] \oplus IV_1 \Leftrightarrow IV \xrightarrow{m_1} (x_1, y_1)$ として, これを初期ブロックと呼ぶ.

$IV \xrightarrow{\varepsilon} (IV, IV)$ をダミーブロックと呼び, $IV \xrightarrow{\varepsilon} (IV, IV) \xrightarrow{m_1} (x_1, y_1) \Leftrightarrow IV \xrightarrow{m_1} (x_1, y_1)$ とする.

定義 27 (MSponge パス) $IV \xrightarrow{m_1} (x_1, y_1)$, $(x_1, y_1) \xrightarrow{m_2} (x_2, y_2)$, ..., $(x_{j-1}, y_{j-1}) \xrightarrow{m_j} (x_j, y_j) \Leftrightarrow IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_j} (x_j, y_j)$ or $IV \xrightarrow{m_1 \| m_2 \| \ldots \| m_j} (x_j, y_j)$ として, これを MSponge パスと呼ぶことにする. そして, T_p を MSponge パスの集合とする.

定義 28 (完全な MSponge パス) $IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_\ell} (x_\ell, y_\ell) \xrightarrow{m_{\ell+1}} \cdots \xrightarrow{m_i} (x_i, y_i) \wedge m_{\ell+1} = m_{\ell+2} = \cdots = m_i = 0^r \wedge \mathsf{pad}^{-1}(m_1 \| m_2 \| \cdots \| m_\ell) = m^* \neq \perp$ となる MSponge パスを完全な MSponge パスと呼ぶ. そして, T_p^* を完全な MSponge パスの集合とする.

定義 29 (MSponge の関係) 定義 28 で記載されている完全な MSponge パスに対して, $L(n, m^*) = (y_\ell[1, r'] \| y_{\ell+1}[1, r'] \| \cdots \| y_i[1, r'])[1, n]$ となる関係を MSponge の関係と呼ぶ.

定義 30 (R クエリの種類) $y = R_F(x)$ によって定義された (x, y) を $\overrightarrow{(x, y)}$, $x = R_I(y)$ によって定義された (x, y) を $\overleftarrow{(x, y)}$ と書く.

定義 31 (パスの定義の順番) パスの定義の順番を下付の数字で表現する. 例えば, $(x_1, y_1) \xrightarrow{m_2} (x_2, y_2) \wedge (x_1, y_1)$ が定義された後に (x_2, y_2) が定義される場合, $(x_1, y_1)_1 \xrightarrow{m_2} (x_2, y_2)_2$ と書く.

S が対応すべきケース

5.2.3 節では入力長が 2 ブロック, 出力長が 1 ブロックの $\mathsf{MSponge}^P$ に限定して, $\mathcal{D}$ が識別できないための S が対応すべきケースとして, ケース 1 とケース 2 の 2 ケースがあることを説明した. 2 つのケースをもう一度以下に記載しておく.

- ケース 1 (R オラクルに対応するケース): Real World では $(R_F, R_I) = (P, P^{-1})$ なので, $S = (S_F, S_I)$ は (P, P^{-1}) をシミュレートする.
- ケース 2 (L オラクルに対応するケース): Real World は $(L, R_F, R_I) = (\mathsf{MSponge}^P, P, P^{-1})$ で, $\mathsf{MSponge}^P$ は L へのクエリに対して, MSponge の構造に従って P にアクセスしてレスポンスを計算しているため, 完全な MSponge パス

は MSponge の関係を満たす．よって, Ideal World で, S は, 完全な MSponge パスが MSponge の関係を満たすようにシミュレートしなければならない．

本節では, 前節で述べたこの 2 つのケースのアイディアを用いて仕様通りの MSponge の構造に対応した S を構成する．前節で述べたとおり, ケース 1 とケース 2-1 のケース a とケース 2-2 のケース a111 に対応する S が構成できれば, $\mathcal{D}$ は無視できる確率を除いて Real World と Ideal World を識別できないため, これらのケースを考慮に入れた S を定義する．

S の手続き

P をシミュレートする S_F と P^{-1} をシミュレートする S_I を図 5.4 に定義する．

ケース 1 に対応した手続き

まず, ケース 1 に対応した S の手続きとして, 初出のクエリに対して, 出力がランダムになるように定義する．また, S のクエリレスポンスを記録するテーブル T_S を導入して, 繰り返しきたクエリに対してこのテーブルを参照して, 過去の値と同じ値を出力するようにする．この手続きは S_F のステップ 1 で行う．

ケース 2 に対応した手続き

次に, ケース 2 に対応した S の手続きを考える．ケース 2-1 のケース a とケース 2-2 のケース a111 の仕様通りの MSponge に対応するケースは, $y_i[r+1,t] \neq IV_2$ $(1 \leq i \leq j+l-1)$, $\mathsf{pad}^{-1}(m_1\|m_2\|\cdots\|m_j) = m^* \neq \perp$ となる完全な MSponge パス $IV \xrightarrow{m_1} \overrightarrow{(x_1,y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_j} \overrightarrow{(x_j,y_j)}_j \xrightarrow{0^r} \cdots \xrightarrow{0^r} \overrightarrow{(x_{j+l-1},y_{j+l-1})}_{j+l-1}$ となるケースで, このパスに対して, $\mathcal{RO}(m^*)[1,n] = (y_j[1,r']\|y_{j+1}[1,r']\|\cdots\|y_{j+l-1}[1,r'])[1,n]$ となるように S を定義する．S は, MSponge パスをテーブル path で管理して, このテーブルを使って MSponge の関係を満たすようにする．

具体的な手続きは, ステップ 2〜11 で行う．直観的には, $(x_j,y_j),\ldots,(x_{j+l-1},y_{j+l-1})$ が定義される場合, すでにその前のブロックは定義されているため, S_F は $m_1,\ldots,m_j$ を復元することができて, $\mathsf{pad}^{-1}(m_1\|m_2\|\cdots\|m_j) = m^*$ として, $\mathcal{RO}(m^*)[1,n] = (y_j[1,r']\|y_{j+1}[1,r']\|\cdots\|y_{j+l-1}[1,r'])[1,n]$ となるように $y_j, y_{j+1}, \cdots, y_{j+l-1}$ を定義することで MSponge の関係を満たすことができる．

ステップ 2〜11 の詳細な手続きは, クエリ x を手掛かりに MSponge パス $IV \xrightarrow{m} (x^*, y^*\|x[r+1,t])$ を見つけて, m を後半 $0^{r'}$ のブロックとそれ以外のメッセージ m_0 に分け

Initialization

1. $T_S \leftarrow \emptyset$
2. $\mathsf{path} \leftarrow \{IV \xrightarrow{\varepsilon} (IV, IV)\}$

$S_F(x)$

1. **if** $\exists (x, y) \in T_S$ **then return** y
2. **if** $\exists (IV \xrightarrow{m} (x^*, y^* \| x[r+1, t]) \in \mathsf{path}$ **then**
3. parse $m \| (y^* \oplus x[1, r])$ into m_0 and $0^{\rho r'}$
   s.t. the last block of m_0 is not $0^{r'}$
4. $y[r'+1, t] \xleftarrow{\$} \{0,1\}^{c'}$
5. **if** $\mathsf{pad}^{-1}(m_0) = m^* \neq \perp$ **then**
6. $z \leftarrow \mathcal{RO}(m^*)$
7. assign $\rho + 1$-th r'-bit block of z to $y[1, r']$
8. **else**
9. $y[1, r'] \xleftarrow{\$} \{0,1\}^{r'}$
10. **endif**
11. insert $IV \xrightarrow{m} (x^*, y^* \| x[r+1, t]) \xrightarrow{y^* \oplus x_a} (x, y)$ to path
12. **else**
13. $y \xleftarrow{\$} \{0,1\}^t$
14. **endif**
15. insert (x, y) to T_S
16. **return** y

$S_I(y)$

1. **if** $\exists (x, y) \in T_S$ **then return** x
2. **else**
   (a) $x \xleftarrow{\$} \{0,1\}^t$
   (b) insert (x, y) to T_S
   (c) **return** x

図.5.4 Simulator

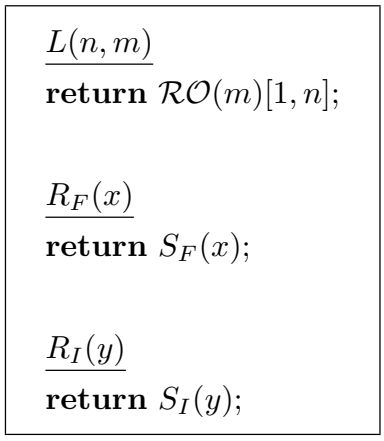

図.5.5 Game 1 (Ideal World)

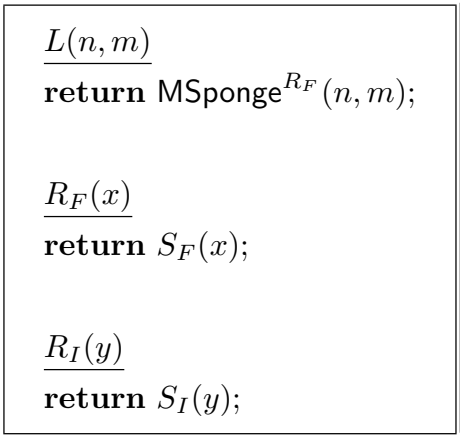

図.5.6 Game 2

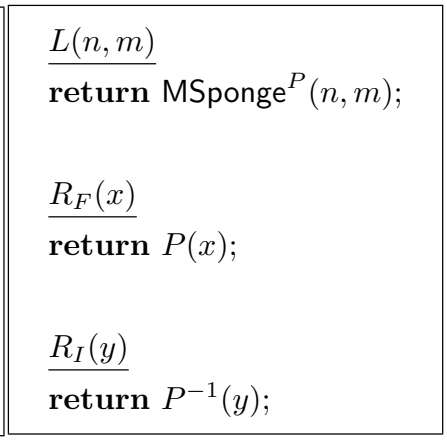

図.5.7 Game 3 (Real World)

る．この MSponge パスは $IV \xrightarrow{m_0} (s, w_1) \xrightarrow{0^{\rho r'}} (x^*, y^* \| x[r+1,t])$ と読みかえることができて, この MSponge パスが完全な MSponge パス, すなわち, $\mathsf{pad}^{-1}(m_0) = m^* \neq \perp$ ならば MSponge の関係を満たすために $\mathcal{RO}(m^*)$ を用いて入力 x の出力 y を定義し, それ以外では y をランダムに選ぶ. そして, ここで定義された入出力ペア (x, y) を用いて MSponge パスを更新し, path に記録しておく.

5.2.5 定理 5.2 の証明

本章では定理 5.2 の証明を図 5.5, 5.6 と 5.7 に記載されている 3 つのゲーム用いて証明する．各々のゲームで, 識別者 $\mathcal{D}$ は 3 つのオラクル (L, R_F, R_I) にアクセスすることができる. Game 1 は Ideal World で Game 3 は Real World である. G_i を $\mathcal{D}$ がゲーム i で 1 を出力するイベントとすると, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MSponge}^P,\mathcal{RO},S}(\mathcal{A}) = |\Pr[G_1] - \Pr[G_3]| \leq |\Pr[G_1] - \Pr[G_2]| + |\Pr[G_2] - \Pr[G_3]|.$$

以下, それぞれの確率の差を評価する.

Game 1 ⇒ Game 2

Game 1 と Game 2 の差を評価する．この評価をするために, Game 1 と Game 2 の中間のゲームにあたる Game 1.5 を以下に定義する. Game 1.5 では, L を次のように定義する. L へのクエリ (n, m) に対して, $\mathsf{MSponge}^{R_F}(n, m)$ を計算する. そして, $\mathcal{RO}(m)[1, n]$ を返信する. またた, Game 1.5 では $(R_F, R_I) = (S_F, S_I)$ である.

以下では, まず, bad イベント Bad を定義して, Bad が起こらない限り, $\mathcal{D}$ は Game 1 と

Game 1.5 を識別できないことを示す. 次に, Bad が起こらない限り, $\mathcal{D}$ は Game 1.5 と Game 2 を識別できないことを示す. よって, Bad が起こらない限り, $\mathcal{D}$ は Game 1 と Game 2 を識別できないことが言えて, Game 1 と Game 2 の差は Bad が起こる確率で抑えることができる. 最後に, Bad が起こる確率を評価する.

以下に bad イベント Bad を定義する. $\mathsf{Bad} = \mathsf{Bad}_1 \vee \mathsf{Bad}_2 \vee \mathsf{Bad}_3 \vee \mathsf{Bad}_4$ とする. $\mathsf{Bad}_1, \mathsf{Bad}_2, \mathsf{Bad}_3, \mathsf{Bad}_4$ の定義は以下の通りである.[*1]

- Bad_1: $\exists \overrightarrow{(x,y)}_2 \xrightarrow{m} (x',y')_1 \in T_b$
- Bad_2: $\exists (x,y)_1 \xrightarrow{m} \overleftarrow{(x',y')}_2 \in T_b$
- Bad_3: $\exists IV \xrightarrow{m} \overleftarrow{(x,y)} \in T_b$
- Bad_4: $\exists \left(IV \xrightarrow{m} (x,y)\right), \left(IV \xrightarrow{m'} (x',y')\right) \in T_p$ s.t. $y[r+1,t] = y'[r+1,t]$.

次に, 以下の 2 つの補題を示す.

補題 17 Bad が起こらなければ $\mathcal{D}$ は Game 1 と Game 1.5 を識別できない.

証明. Game 1 と Game 1.5 の差は, Game 1 では, L は R_F にクエリをしないが, Game 1.5 では, L は R_F に MSponge の構造に依存したクエリをする. よって, この L によるクエリが Game 1 と Game 1.5 を識別するためのアドバンテージを $\mathcal{D}$ に与えないことを示す. このクエリは完全な MSponge パスに注目すればよい.

まず, Game 1 と Game 1.5 で, $\mathsf{Bad} = \mathsf{false}$ ならば,

$$\mathsf{pad}^{-1}(m_1 \| \cdots \| m_j) = m^* \text{となる}$$
$$IV \xrightarrow{m_1} (x_1,y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_j} (x_j,y_j) \xrightarrow{0^r} \cdots \xrightarrow{0^r} (x_{j+l-1}, y_{j+l-1}) \in T_p^*$$
$$\Rightarrow \mathcal{RO}(m^*)[1,n] = (y_j[1,r'] \| y_{j+1}[1,r'] \| \cdots \| y_{j+l-1}[1,r'])[1,n]$$

[*1] 5.2.3 節はシミュレートに失敗するケースとしてケース 1, ケース 2-1 の b, ケース 2-2 の a112, a12, a131, a132, a2, b を述べた. ここでは, これらのケースと Bad_1, Bad_2, Bad_3, Bad_4 の対応関係を示す.

- Bad_1 はケース 2-2 の a112, a132 に対応したイベントである. a112 は $\overrightarrow{(x_1,y_1)}_2 \xrightarrow{m_2} \overrightarrow{(x_2,y_2)}_1$ となるイベント, a132 は $\overrightarrow{(x_1,y_1)}_2 \xrightarrow{m_2} \overleftarrow{(x_2,y_2)}_1$ となるイベントで, 共に Bad_1 がカバーするイベントである.
- Bad_2 はケース 2-2 の a131 に対応したイベントである. a131 は $\overrightarrow{(x_1,y_1)}_1 \xrightarrow{m_2} \overleftarrow{(x_2,y_2)}_2$ となるイベントで, Bad_2 はこのイベントをカバーする.
- Bad_3 はケース 2-1 の b, ケース 2-2 の b に対応するイベントである. ケース 2-1 の b とケース 2-2 の b は共に $IV \xrightarrow{m_2} \overleftarrow{(x_1,y_1)}_1$ となるイベントで, Bad_3 はこのイベントをカバーする.
- Bad_4 はケース 2-2 の a12, a2 に対応するイベントである. ケース 2-2 の a12, a2 共に MSponge パスの衝突に関するイベントで, Bad_4 はこれらのイベントをカバーする.

ここで, ケース 1 は Game 1 と Game 2 で差が出るイベントではないことに注意されたい. そして, ケース 1 は Game 2 と Game 3 の差を評価するときに考えるイベントである.

となることを示す．これにより, $\mathsf{Bad} = \mathsf{false}$ ならば Game 1 と Game 1.5 で任意の完全な MSponge パスは同じ関係を満たすこととなるため, Game 1.5 で行われる L が行う MSponge 構造に依存した R_F へのクエリは $\mathcal{D}$ の振る舞いに影響を与えない．よって, Bad が起こらなければ $\mathcal{D}$ は Game 1 と Game 1.5 を識別できないことが言える．

上記のことを示すために, 初めに, $\mathsf{Bad} = \mathsf{false}$ ならば以下を満たすことを数学的帰納法を用いて証明する．

$$IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_j} (x_i, y_i) \in T_p \Rightarrow IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_j} \overrightarrow{(x_i, y_i)}_i.$$

$i = 1$ の場合, Bad_3 は起こらないため, $IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1$ となる．次に, $i = s$ の場合, 以下が成り立つと仮定する．

$$IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_s} \overrightarrow{(x_s, y_s)}_s$$

まず, Bad_1 は起こらないため以下が成り立つ．

$$IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_s} \overrightarrow{(x_s, y_s)}_s \xrightarrow{m_{s+1}} (x_{s+1}, y_{s+1})_{s+1}$$

さらに, Bad_2 は起こらないため以下が成り立つ．

$$IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_s} \overrightarrow{(x_s, y_s)}_s \xrightarrow{m_{s+1}} \overrightarrow{(x_{s+1}, y_{s+1})}_{s+1}$$

ここで, $\mathsf{Bad}_4 = \mathsf{false}$ なので, $\forall i \in \{1, \ldots, j+l-1\}$ に対して, x_i が R_F にクエリされた時に, $y'[r+1, t] = y_{i-1}[r+1, t]$ となる上記とは異なる MSponge パス $IV \xrightarrow{m'} (x', y')$ は存在しない．よって, S_F の手続きより, x_i が R_F にクエリされた際に, path 内に記録されている x_i とつながる MSponge パスは以下のパスのみである．

$$IV \xrightarrow{m_1} \overrightarrow{(x_1, y_1)}_1 \xrightarrow{m_2} \cdots \xrightarrow{m_i} \overrightarrow{(x_{i-1}, y_{i-1})}_{i-1} \in \mathsf{path}$$

ここで, $j+1 \leq i$ の場合, $m_i = 0^r$ であることに注意されたい．よって, $\forall i \in \{j, \ldots, j+l-1\}$ に対して, R_F へのクエリ x_i の出力 y_i の $y_i[r'+1, t]$ は, S_F のステップ 6,7 の手続きで定義されるため, 以下が成り立つ．

$$\begin{aligned}
&\mathsf{pad}^{-1}(m_1 \| \cdots \| m_l) = m^* \text{となる} \\
&IV \xrightarrow{m_1} (x_1, y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_j} (x_j, y_j) \xrightarrow{0^r} \cdots \xrightarrow{0^r} (x_{j+l-1}, y_{j+l-1}) \in T_p^* \\
&\Rightarrow \mathcal{RO}(m^*)[1, n] = (y_j[1, r'] \| y_{j+1}[1, r'] \| \cdots \| y_{j+l-1}[1, r'])[1, n].
\end{aligned}$$

上記の議論は, Game 1 と Game 1.5 両方のケースで成り立つ．以上より, Bad が起こらなければ $\mathcal{D}$ は Game 1 と Game 1.5 を識別できない．

■

補題 18 Bad が起こらなければ $\mathcal{D}$ は Game 1.5 と Game 2 を識別できない.

証明. Game 1.5 と Game 2 の差は, Game 1.5 では $L(n,m) = \mathcal{RO}(m)[1,n]$ であるが, Game 2 では $L(n,m) = \mathsf{MSponge}^{S_F}(n,m)$ である点である.

ここで, 補題 17 の証明の中で示した, $\mathsf{Bad} = \mathsf{false}$ ならば,

$$\begin{aligned}
&\mathsf{pad}^{-1}(m_1\|\cdots\|m_l) = m^* \text{となる} \\
&IV \xrightarrow{m_1} (x_1,y_1) \xrightarrow{m_2} \cdots \xrightarrow{m_j} (x_j,y_j) \xrightarrow{0^r} \cdots \xrightarrow{0^r} (x_{j+l-1},y_{j+l-1}) \in T_p^* \\
&\Rightarrow \mathcal{RO}(m^*)[1,n] = (y_j[1,r']\|y_{j+1}[1,r']\|\cdots\|y_{j+l-1}[1,r'])[1,n]
\end{aligned}$$

となることは, Game 2 に対しても成り立つため, Game 2 でも, $\mathsf{MSponge}^{R_F}(n,m) = \mathcal{RO}(m)[1,n]$ となる.

以上より, Bad が起こらなければ $\mathcal{D}$ は Game 1.5 と Game 2 を識別できない.

■

上記の 2 つの補題を合わせると以下の系が成り立つ.

系 3 Bad が起こらなければ $\mathcal{D}$ は Game 1 と Game 2 を識別できない.

$j \in \{1,2\}$ に対して, $\mathsf{Bad}j \Leftrightarrow$ Game j で $\mathsf{Bad} = \mathsf{true}$ となるイベントとすると, 上記の系より, $\Pr[G_1|\neg\mathsf{Bad1}] = \Pr[G_2|\neg\mathsf{Bad2}]$ となる. この事実を利用するとが成り立つ.

$$\begin{aligned}
|\Pr[G_1] - \Pr[G_2]| \leq &|\Pr[G_1|\mathsf{Bad1}]\Pr[\mathsf{Bad1}] + \Pr[G_1|\neg\mathsf{Bad1}]\Pr[\neg\mathsf{Bad1}] \\
&- (\Pr[G_2|\mathsf{Bad2}]\Pr[\mathsf{Bad2}] + \Pr[G_2|\neg\mathsf{Bad2}]\Pr[\neg\mathsf{Bad2}])| \\
\leq &|\Pr[G_1|\neg\mathsf{Bad1}](\Pr[\mathsf{Bad2}] - \Pr[\mathsf{Bad1}]) \\
&+ (\Pr[G_1|\mathsf{Bad1}]\Pr[\mathsf{Bad1}] - \Pr[G_2|\mathsf{Bad2}]\Pr[\mathsf{Bad2}])| \\
\leq &|\Pr[\mathsf{Bad2}](\Pr[G_1|\neg\mathsf{Bad1}] - \Pr[G_2|\mathsf{Bad2}]) \\
&- \Pr[\mathsf{Bad1}](\Pr[G_1|\neg\mathsf{Bad1}] - \Pr[G_1|\mathsf{Bad1}])|
\end{aligned}$$

$\Pr[G_1|\neg\mathsf{Bad1}] = p$, $\Pr[G_1|\mathsf{Bad1}] = p1$, $\Pr[G_2|\mathsf{Bad2}] = p2$ とおくと,

$$\begin{aligned}
&|\Pr[\mathsf{Bad2}](\Pr[G_1|\neg\mathsf{Bad1}] - \Pr[G_2|\mathsf{Bad2}]) - \Pr[\mathsf{Bad1}](\Pr[G_1|\neg\mathsf{Bad1}] - \Pr[G_1|\mathsf{Bad1}])| \\
&\quad = |\Pr[\mathsf{Bad2}](p - p2) - \Pr[\mathsf{Bad1}](p - p1)|
\end{aligned}$$

そして, $p2 \geq p1$ の場合,

- $p \geq p2 \geq p1$ の場合:

$$|\Pr[\mathsf{Bad2}](p-p2) - \Pr[\mathsf{Bad1}](p-p1)| \leq \max\{\Pr[\mathsf{Bad1}], \Pr[\mathsf{Bad2}]\}$$

- $p2 \geq p \geq p1$ の場合:

$$\begin{aligned}|\Pr[\mathsf{Bad2}](p-p2) - \Pr[\mathsf{Bad1}](p-p1)| &= \Pr[\mathsf{Bad2}](p-p1) - \Pr[\mathsf{Bad1}](p2-p) \\ &\leq \Pr[\mathsf{Bad2}]p + \Pr[\mathsf{Bad1}](1-p) \\ &\leq \max\{\Pr[\mathsf{Bad1}], \Pr[\mathsf{Bad2}]\}\end{aligned}$$

- $p2 \geq p1 \geq p$ の場合:

$$|\Pr[\mathsf{Bad2}](p-p2) - \Pr[\mathsf{Bad1}](p-p1)| \leq \max\{\Pr[\mathsf{Bad1}], \Pr[\mathsf{Bad2}]\}$$

そして, $p2 \leq p1$ の場合は $p2 \geq p1$ の場合と同様に評価できる.

以上より, $|\Pr[G_1] - \Pr[G_2]| \leq \max\{\Pr[\mathsf{Bad1}], \Pr[\mathsf{Bad2}]\}$ となる.

以下, $\Pr[\mathsf{Bad1}], \Pr[\mathsf{Bad2}]$ を評価する.

まず, $\Pr[\mathsf{Bad1}]$ を評価する. $\mathsf{Bad1} = \mathsf{Bad1}_1 \vee \mathsf{Bad1}_2 \vee \mathsf{Bad1}_3 \vee \mathsf{Bad1}_4$ なので, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad1}] \leq \Pr[\mathsf{Bad1}_1] + \Pr[\mathsf{Bad1}_2] + \Pr[\mathsf{Bad1}_3] + \Pr[\mathsf{Bad1}_4]$$

以下, これらの確率を評価する.

$\Pr[\mathsf{Bad1}_1]$ の評価: まず, イベント $\mathbf{MColl}_d$ を定義する. $z_1, \ldots, z_{q_L}$ を $\mathcal{D}$ から $\mathcal{RO}$ へのクエリに対するすべてのレスポンスとする. そして, $z_{i,j} := z_i[(j-1)r'+1, jr']$ とする. すなわち, $z_{i,j}$ は r' bit を 1 ブロックの bit とした場合の z_i の j 番目のブロックである.

- $\mathbf{MColl}_d$: $\exists j^*, \exists i_1, \ldots, i_d$ s.t. $z_{i_1,j^*}[r+1, r'] = \ldots = z_{i_d,j^*}[r+1, r']$.

$\mathbf{MColl}_d$ を用いると, 以下の式を得る.

$$\begin{aligned}\Pr[\mathsf{Bad1}_1] =& \Pr[\mathsf{Bad1}_1 \wedge \mathbf{MColl}_d] + \Pr[\mathsf{Bad1}_1 \wedge \neg\mathbf{MColl}_d] \\ \leq& \Pr[\mathbf{MColl}_d] + \Pr[\mathsf{Bad1}_1 | \neg\mathbf{MColl}_d]\end{aligned}$$

ここで, 出力のブロック数は最大 N ブロックなので, $\Pr[\mathbf{MColl}_d] \leq N p_{\mathsf{mcoll}}(q_L, d, c^*)$ となる.

$\Pr[\mathsf{Bad1}_1 | \neg\mathbf{MColl}_d]$ を評価する. $\exists \overrightarrow{(x,y)}_2 \xrightarrow{m} (x', y')_1 \in T_p$ となる確率は次のとおりである. $y[r+1, r']$ は, S のステップ 6 と 7 から $\mathcal{RO}$ にあらかじめ問い合わせることで, 確率 1 で

$y[r+1,r'] = x'[r+1,r']$ とできる. 残りの c' bit$y[r'+1,t]$ は $x'[r'+1,t]$ とは独立にランダムに選ばれるため, $(x,y),(x',y')$ を固定したときに $y[r'+1,t] = x'[r'+1,t]$ となる確率は $1/2^{c'}$ である. そして, $\mathbf{MColl}_d = \mathsf{false}$ なので, 1 つの y に対して, x' の候補は高々 $d-1$ 個である. (x,y) は q_F 個存在するため, $\exists \overrightarrow{(x,y)}_2 \xrightarrow{m'} (x',y')_1 \in T_b$ となる確率は $(d-1)q_F/2^{c'}$ となる.

$\Pr[\mathsf{Bad1}_2]$ の評価: $\exists (x,y)_1 \xrightarrow{m} \overleftarrow{(x',y')}_2 \in T_b$ となる確率は, x' は y とは独立にランダムに選ばれ, (x,y) は $q_F + q_I$ 個存在し, R_I へのクエリ回数は q_I なので, $q_I(q_F+q_I)/2^c$ となる.

$\Pr[\mathsf{Bad1}_3]$ の評価: $\exists IV \xrightarrow{m} \overleftarrow{(x,y)} \in T_b$ となる確率は, x はランダムに選ばれ, R_I へのクエリ回数は q_I なので, $q_I/2^c$ である.

$\Pr[\mathsf{Bad1}_4]$ の評価: $\exists \left(IV \xrightarrow{m} (x,y)\right), \left(IV \xrightarrow{m^*} (x^*,y^*)\right) \in \mathsf{path}$ s.t. $y[r+1,t] = y^*[r+1,t]$ となる確率は, y, y^* は独立にランダムに選ばれ, $(x,y),(x',y')$ はそれぞれ q_F+1 個存在するため, 誕生日解析を適用すると, $q_F(q_F+1)/2^{c+1}$ となる.

以上より, 以下の式が成り立つ.

$$\begin{aligned}\Pr[\mathsf{Bad}_1] \leq & \frac{(q_F+q_I)^2 + q_I^2 + q_F + q_I}{2^{c+1}} + \frac{(d-1)q_F}{2^{c'}} + \Pr[\mathbf{MColl}_d] \\ \leq & \frac{(d-1)q}{2^{c'}} + \frac{q(q-1)}{2^c} + N p_{\mathsf{mcoll}}(q_L, d, c^*)\end{aligned}$$

最後に, $\Pr[\mathsf{Bad2}]$ を評価する. $\Pr[\mathsf{Bad1}]$ と同様に, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad2}] \leq \Pr[\mathsf{Bad2}_1] + \Pr[\mathsf{Bad2}_2] + \Pr[\mathsf{Bad2}_3] + \Pr[\mathsf{Bad2}_4]$$

以下, これらの確率を評価する.

- $\Pr[\mathsf{Bad2}_1]$ の評価: $\exists \overrightarrow{(x,y)}_2 \xrightarrow{m} (x',y')_1 \in T_b$ となる確率は, y は x' とは独立にランダムに選ばれ, (x,y) は $Nq_L + q_F$ 個, (x',y') は σ 個存在するため, $\sigma(Nq_L+q_F)/2^c$ となる.
- $\Pr[\mathsf{Bad2}_2]$ の評価: $\exists (x,y)_1 \xrightarrow{m} \overleftarrow{(x',y')}_2 \in T_b$ となる確率は, x' は y とは独立にランダムに選ばれ, (x,y) は σ 個, (x',y') は q_I 個存在するため, $\sigma q_I/2^c$ となる.
- $\Pr[\mathsf{Bad2}_3]$ の評価: $\exists IV \xrightarrow{m} \overleftarrow{(x,y)} \in T_b$ となる確率は, x はランダムに選ばれるため, $q_I/2^c$ となる.

- $\Pr[\mathsf{Bad2}_4]$ の評価: $\exists \left(IV \xrightarrow{m} (x,y)\right), \left(IV \xrightarrow{m^*} (x^*,y^*)\right) \in T_p$ s.t. $y[r+1,t] = y^*[r+1,t]$ となる確率は, y, y^* は独立にランダムに選ばれ, $(x,y),(x',y')$ はそれぞれ σ 個存在するため, 誕生日解析を適用すると, $\sigma(\sigma-1)/2^{c+1}$ となる.

以上より, 以下の式が成り立つ.

$$\Pr[\mathsf{Bad2}] \leq \frac{\sigma(Nq_L+q_F)+\sigma q_I+q_I+0.5\sigma(\sigma-1)}{2^c} \leq \frac{\sigma^2}{2^{c-1}}.$$

以上より, 以下の式が成り立つ.

$$\Pr[G_1]-\Pr[G_2] \leq \max\{\frac{(d-1)q}{2^{c'}}+\frac{q(q-1)}{2^c}+Np_{\mathsf{mcoll}}(q_L,d,c^*), \frac{\sigma^2}{2^{c-1}}\}.$$

Game 2 ⇒ Game 3. Game 2 では $(R_F,R_I)=(S_F,S_I)$ で S_F と S_I の出力はランダムに選ばれるのに対し, Game 3 では $(R_F,R_I)=(P,P^{-1})$ である. よって, PRF/PRP switching lemma [10] (詳しくは付録を参照されたい) より, 以下の式が成り立つ.

$$|\Pr[G_2]-\Pr[G_3]| \leq \frac{\sigma(\sigma-1)}{2^{t+1}}.$$

以上より, 以下の式が成り立つ.

$$\begin{aligned}\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MSponge}^P,\mathcal{RO},S}(\mathcal{A}) \leq & |\Pr[G_1]-\Pr[G_2]|+|\Pr[G_2]-\Pr[G_3]| \\ \leq & \max\{\frac{(d-1)q}{2^{c'}}+\frac{q(q-1)}{2^c}+Np_{\mathsf{mcoll}}(q_L,d,c^*), \frac{\sigma^2}{2^{c-1}}\} \\ & +\frac{\sigma(\sigma-1)}{2^{t+1}}.\end{aligned}$$

5.3 証明の最適性

本節で, 定理 5.2 で与えたバウンドの最適性を示す.

5.3.1 MSponge と $\mathcal{RO}$ に対する識別攻撃

以下, (L,R_F,R_I) にアクセスして, Real World である $(L,R_F,R_I)=(\mathsf{MSponge}^P,P,P^{-1})$ と Ideal World である $(L,R_F,R_I)=(\mathsf{MSponge}^P,S_F,S_I)$ を識別する識別者 $\mathcal{D}$ を構成する. $\mathcal{D}$ が行う L へのクエリのうち L の出力長は全て 1 ブロック分すなわち r' bit となるようにする. L の出力長は全て r' bit となるため, 以下の説明では, L へのクエリで L の出力長は省略して議論する.

1. まず, $\mathsf{pad}(m_1)$ が r bit に収まるようにメッセージ m_1 を選ぶ. 次に, $z \leftarrow L(m_1)$ とする. この操作を, $z[r+1, r']$ で d-multi-collision が起こるまで行う. $m_{1,1}, \ldots, m_{1,d}$ を d-multi-collision が発生したメッセージとして, これらの出力値をそれぞれ $z_{1,1}, \ldots, z_{1,d}$ とする. $z_{1,1}[r+1, r'] = \cdots = z_{1,d}[r+1, r']$ となる. $m^*_{1,i} := \mathsf{pad}(m_{1,i})$ $(i \in \{1, \ldots, d\})$, $z^*_1 := z_{1,1}[r+1, r']$ とする.
2. 次に, $2^{c'}/d$ 個の異なる c' bit の値 $z'_{1,j}$ を選んで, それぞれの j について $x_{2,j} := 0^r \| z^*_1 \| z'_{1,j}$ として, $y_{2,j} \leftarrow R_F(x_{2,j})$ として, $x_{3,j} \leftarrow (pad_3 \| 0^c) \oplus y_{2,j}$, $y_{3,j} \leftarrow R_F(x_{3,j})$ とする. pad_3 は pad によって 2 ブロックのメッセージにパディングされる値とする.
3. ステップ 1 で定義した各々の, $z_{1,i}$ に対して, $m_{2,i} := z_{1,i}[1, r]$ として, $z_{3,i} \leftarrow L(m^*_{1,i} \| m_{2,i})$ とする. ここで, もし $z_{3,i} = y_{3,j}[1, r']$ となる j が存在するならば, 以下の操作を実行する.
   (a) $y_{1,i} \leftarrow (m_{2,i} \| 0^c) \oplus x_{2,j}$ とする.
   (b) r-bit のメッセージ m_2 を選んで, $x_2 \leftarrow (m_2 \| 0^c \oplus y_{1,i})$, $y_2 \leftarrow R_F(x_2)$, $y_3 \leftarrow R_F((pad_3 \| 0^c) \oplus y_2)$ とする.
   (c) $z_3 \leftarrow L(m^*_{1,i} \| m_2)$ とする.
   (d) もし $z_3 = y_3[1, r']$ ならば 1 を出力する.
4. 0 を出力する.

5.3.1.1 クエリ回数と識別確率.

ステップ 1 で, d-multi-collision を確率 $1/2$ で起こすためのクエリ回数は $q_{\mathsf{mcoll}}(d, c^*)$ である. ここで必要なメモリは d-multi-collision を見つけるための $q_{\mathsf{mcoll}}(d, c^*)$ 個の内部状態と d 個の $(m_{1,i}, z_{1,i})$ を記録するための領域である. ステップ 2 で, $2 \times \lceil 2^{c'}/d \rceil$ 回の R_F へのクエリを行う. ここで, 必要なメモリは $\lceil 2^{c'}/d \rceil$ 個の $(x_{2,j}, y_j[1, r'])$ を記録する分である. ステップ 3 で, 2 ブロック分の L へのクエリを d 回行い, そして, $z_{3,i} = y_{3,j}[1, r']$ の場合, R_F へのクエリを 2 回と L への 2 ブロッククエリを 1 回行う. 以上より, クエリ回数は $q_{\mathsf{mcoll}}(d, c^*) + 2(d+1) + 2 \times \lceil 2^{c'}/d \rceil + 2$ で必要なメモリは $q_{\mathsf{mcoll}}(d, c^*) + \lceil 2^{c'}/d \rceil$ となる.

そして, ステップ 1 の成功確率は $1/2$ で, ステップ 3 の成功確率はおよそ $1 - 1/e$ であることから, 上記の成功確率は $1/2 \cdot (1 - 1/e) \approx 0.316$ となる.

5.3.2 Discussion

定理 5.2 で, $\mathsf{MSponge}^P$ のアドバンテージが以下で抑えられることを証明した.

$$\max\{\frac{(q_I+1)(q_F+q_I)}{2^c}+\frac{(d-1)q_F}{2^{c'}}+Np_{\mathsf{mcoll}}(q_L,d,c^*),\frac{\sigma(\sigma+1)}{2^c}\}+\frac{\sigma(\sigma-1)}{2^{t+1}}.$$

ここで, $q_{\mathsf{mcoll}}(d^*,c^*)=2^{c'}/d^*$ となる d^* を d に代入すると, このバウンドが固定値となるクエリ回数の下界は $\mathcal{O}\left(\min\{2^{c'}/d^*,2^{c/2}\}\right)$ となる.

一方で, 前章で示した識別攻撃から識別するためのクエリ回数の上界は $\mathcal{O}\left(\min\{2^{c'}/d^*,2^{c/2}\}\right)$ となる.

以上より, 定理 5.2 で与えたバウンドから得られる識別計算量の下界は最適である.

5.4 キャパシティサイズ c' について

本章で, $c'\geq c/2+\log_2 c$ ならば, MSponge が $\mathcal{O}(2^{c/2})$ の安全性を担保できることを示す.

MSponge の IFRO のアドバンテージは以下の通りである.

$$\max\{\frac{(q_I+1)(q_F+q_I)}{2^c}+\frac{(d-1)q_F}{2^{c'}}+Np_{\mathsf{mcoll}}(q_L,d,c^*),\frac{\sigma(\sigma+1)}{2^c}\}+\frac{\sigma(\sigma-1)}{2^{t+1}}.$$

ここで, 上記の確率の $(q_I+1)(q_F+q_I)/2^c,\sigma(\sigma+1)/2^c,\sigma(\sigma-1)/2^{t+1}$ は $\mathcal{O}(2^{c/2})$ の安全性を保証することから, 以下その他の確率について議論する.

まず, $p_{\mathsf{mcoll}}(Nq_L,d,c^*)$ について, 以下の式が成り立つ.

$$p_{\mathsf{mcoll}}(Nq_L,d,c^*)\leq 2^{c^*}\binom{Nq_L}{d}\left(\frac{1}{2^{c^*}}\right)^d.$$

ここで, $d=c^*$ とスターリングの近似 (任意の整数 x に対して $x!\leq(x/e)^x$) を用いると以下の式が成り立つ.

$$2^{c^*}\binom{Nq_L}{c^*}\left(\frac{1}{2^{c^*}}\right)^{c^*}\leq 2^{c^*}\left(\frac{eNq_L}{c^*2^{c^*}}\right)^{c^*}\leq\left(\frac{eNq_L}{c^*2^{c^*-1}}\right)^{c^*}$$

また, $\frac{(d-1)q_F}{2^{c'}}$ の項は, $d=c^*$ とすると, $\frac{(c^*-1)q_F}{2^{c'}}$ となる.

ここで, $c'=c/2+\log_2 c$ とすると, $c^*=c/2-\log_2 c$ となり, $c/4\geq\log_2 c$ の場合, 上記の 2 つの項は以下の通りとなる.

$$\left(\frac{eNq_L}{c/4\times 2^{c/2-\log_2 c-1}}\right)^{c/4}\leq\left(\frac{eNq_L}{2^{c/2-3}}\right)^{c/4}\qquad\frac{(c/2-\log_2 c-1)q_F}{2^{c/2+\log_2 c}}\leq\frac{q_F}{2^{c/2+1}}$$

以上より, $c'\geq c/2+\log_2 c$ の場合に MSponge は $\mathcal{O}(2^{c/2})$ の安全性を満たす.

5.5 MSponge*

本章では, MSponge の absorbing step の第 1 ブロックのキャパシティを c_1 bit に変更したときの安全性について議論する. $r_1 = t - c_1$ とする. ここで, $c > c_1$ ならば, 第 1 ブロックで扱えるメッセージ長が長くなり, MSponge を高速化できる.

まず, 5.2.5 章の定理 5.2 の証明で, c_1 の長さに関する bad イベントとして Bad_1 と Bad_4 がある. 定理 5.2 の証明では, $c_1 = c$ のケースについて議論しており, このイベントが起こる確率はそれぞれ Game 1 で $q_F/2^{c_1}, q_I/2^{c_1}$, Game 2 で $\sigma/2^{c_1}, q_I/2^{c_1}$ となる. 以上の議論から $\mathsf{MSponge}^*$ の安全性は以下の通りとなる.

定理 5.2 $c' \leq c$ とする. あるシミュレータ $S = (S_F, S_I)$ が存在して, 任意の識別者 $\mathcal{D}$ に対して以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{MSponge}^{*P},\mathcal{RO},S}(\mathcal{A}) \leq \max\{\frac{q_I(q_F+q_I)}{2^c} + \frac{(d-1)q_F}{2^{c'}} + Np_{\mathsf{mcoll}}(q_L, d, c^*), \frac{\sigma(\sigma+1)}{2^c}\} + \frac{\sigma}{2^{c_1}} + \frac{\sigma(\sigma-1)}{2^{t+1}}.$$

ここで, $\mathcal{D}$ は L, R_F, R_I にそれぞれ q_L, q 回クエリできるとする. Real World では $(L, R_F, R_I) = (\mathsf{MSponge}^{*P}, P, P^{-1})$, Ideal World では $(L, R_F, R_I) = (\mathcal{RO}, S_F, S_I)$ である. N を L へのクエリの入力ブロック数と出力ブロック数の和の最大値とする. d はフリーパラメータで上記のバウンドが最小となるように設定する. $\sigma := Nq_L + q$ とする. そして, S のクエリ回数は最大 q で, S の動作時間は $\mathcal{O}(q)$ である. ♦

そして, 5.3 章の議論と組み合わせると, $d = c^*$, $c' \geq c/2 + \log_2 c$, $c_1 \geq c/2$ の場合, $\mathsf{MSponge}^*$ が $\mathcal{O}(2^{c/2})$ の安全性を満たすことが言える.

第 6 章

IFRO 安全なブロック暗号ベースの倍ブロック長構造の設計

6.1 はじめに

ブロック暗号をベースのハッシュ関数の設計方法は最もよく使われているハッシュ関数の設計方法である. その中でも, PGV 方式 [50] は最もよく使われる手法で, ブロック暗号が理想的なブロック暗号 (Ideal Cipher (IC)) ならば衝突困難性を満たす入出力長が固定の関数 (圧縮関数) となることが証明されている [16, 56]. そして, (Strengthened) Merkle-Damgård (MD) 構造などの衝突困難性を満たす圧縮関数をベースに衝突困難性を満たすハッシュ関数の構造と組み合わせることで, 衝突困難性を満たすブロック暗号ベースのハッシュ関数を得ることができる.

ところで, 既存のブロック暗号 (例えば AES) を用いてハッシュ関数を設計すると, ハッシュ関数とブロック暗号を両方実装する場合において, ブロック暗号の部分のプログラムや回路を共通化できるため, 実装サイズを小さくできるメリットがある. PGV 方式の出力長はブロック暗号の出力長と同じサイズとなるが, 既存の多くのブロック暗号の出力長は 128 bit となるため, 衝突困難性の安全性ビットとして 64 bit 安全性までしか保障できない. 一方で, 現在安全性 bit として 128 bit レベルの安全性が必要とされており, 128 bit レベルの衝突困難性を満たす場合ハッシュ関数の出力長は 256 bit 必要である.

この問題を解決する方法として, 圧縮関数の出力長をブロック暗号のブロック長の倍の長さにする, 倍ブロック長構造が知られている. そして, ブロック暗号の出力長を n bit とした場合, IC モデルで n bit の衝突困難性を満たす倍ブロック長構造が提案されている [20, 42, 34, 37, 47, 38, 39]. 倍ブロック長構造の利点は, AES などのブロック暗号を部品

として使うことができるため,

一方で, 最近, SHA-3 の設計にも取り入れられている, 衝突困難性よりも強い安全性である Indifferentiability from a Random Oracle (IFRO) 安全がハッシュ関数の設計のゴールとなっている [40]. IFRO 安全性は, ランダムオラクル ($\mathcal{RO}$) から部品を理想化したハッシュ関数への置き換えを保証する安全性で [40, 51], ハッシュ関数の定義域拡張構造に欠陥がないことを保証することができる.

倍ブロック長構造の設計では, Gong ら [32] の PBGV 構造と Prefix-free Merkle-Damgård 構造を組み合わせた $n/2$ bit の IFRO 安全性を満たす倍ブロック長構造は提案されているものの, n bit の IFRO 安全性を満たす倍ブロック長構造は提案されておらず, この構造の設計は重要な研究課題である.

本章では, Hirose が提案した衝突困難性を満たすブロック暗号ベースの倍ブロック長構造 [34] を拡張して, IFRO 安全性を満たす初めてのブロック暗号ベースの倍ブロック長構造を提案する. 我々が提案する構造は鍵サイズが $2n$ bit, ブロックサイズが n bit のブロック暗号を用いる構造である. 重要なブロック暗号である AES や Camellia は鍵長が 256 bit, ブロック長が 128 bit となる構造を持ち, このパラメータを満たすため, これらのブロック暗号を部品として使うことができる. また, これらのブロック暗号は多くのシステムで実装されており, また, Hirose 構造 [34], Tandem-DM 構造 [37], Abreast-DM 構造 [37] など既存の倍ブロック長構造もこのサイズを用いているため, このサイズを用いることは適当であるといえる.

6.1.1 関連研究

Chang ら [23] と Hirose ら [35] によって, 出力長が n bit のランダムオラクル圧縮関数から n bit の IFRO 安全なハッシュ関数の構造が提案されている. ここで, ブロック暗号から n bit の IFRO 安全となる圧縮関数が存在すれば, 我々のゴールである n bit ブロックのブロック暗号を用いたときに n bit の IFRO 安全性を満たす倍ブロック長構造を得ることができるが, そのような構造は知られていない.

6.2 準備

6.2.1 Hirose の倍ブロック長圧縮関数構造

我々が提案する倍ブロック長構造は Hirose の圧縮関数構造をベースとする構造である [34]. ここでは Hirose の構造を説明する.

定義 32 (Hirose の倍ブロック長圧縮関数構造) $\mathsf{BC}_{2n,n} = (E, D)$ を鍵長が n bit, ブロック長が $2n$ bit のブロック暗号として, E を暗号化関数, D を復号関数とする. $\mathsf{BC}_{2n,n}$ を用いた Hirose の構造 $\mathsf{Hirose}^E : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^{2n}$ は以下の通りである.

- 入力 : $G_{i-1}, H_{i-1}, M_i \in \{0,1\}^n$
- 出力 : $G_i, H_i \in \{0,1\}^n$
- 以下の方法で $(G_i, H_i) = \mathsf{Hirose}^E(G_{i-1}||H_{i-1}, M_i)$ を計算する.

$$G_i = G_{i-1} \oplus E(H_{i-1}||M_i, G_{i-1}) \tag{6.1}$$
$$H_i = C \oplus G_{i-1} \oplus E(H_{i-1}||M_i, G_{i-1} \oplus C,) \tag{6.2}$$

6.1 の手続きを第 1 ブロック, 6.2 の手続きを第 2 ブロックと呼ぶことにする.

6.2.2 Preimage Awareness (PrA) [26, 27].

我々が提案する倍ブロック長構造の安全性証明は Dodis らが提案した Preimage Awareness (PrA) のフレームワークを用いて証明する. ここでは, このフレームワークの概要を本章で必要な部分のみ切り出して説明する.

P を理想的なプリミティブとして, H^{P} を P を用いたハッシュ関数とする. そして, H^{P} に対する PrA の安全性ゲームを以下のように定義する.

$\underline{\mathsf{Exp}^{\mathsf{pra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E},\mathcal{A}}}$	$\underline{\textbf{oracle } \mathsf{h}(m)}$	$\underline{\textbf{oracle } \mathsf{Ex}(z)}$
$x \leftarrow \mathcal{A}^{\mathsf{h},\mathsf{Ex}}$;	$y \leftarrow \mathsf{P}(x)$;	$\mathsf{Q}[z] \leftarrow 1$;
$z \leftarrow \mathsf{H}^{\mathsf{P}}(x)$;	$\alpha \xleftarrow{\cup} (x, y)$;	$\mathsf{V}[z] \leftarrow \mathcal{E}(z, \alpha)$;
Ret $(x \neq \mathsf{V}[z] \wedge \mathsf{Q}[z] = 1)$;	Ret y;	Ret $\mathsf{V}[z]$;

ここで, $\mathcal{A}$ を PrA 攻撃者として, この攻撃者は 2 つのオラクル h と Ex にアクセスることができる. オラクル h はクエリ x に対して, $\mathsf{P}(x)$ の出力値 y を返信して, テーブル α に (x, y) を記録するオラクル. エクストラクションオラクル Ex はクエリ z に対して, エクストラクタ $\mathcal{E}(z, \alpha)$ の出力値を出力するオラクルである. ここで, Q は初期状態として全て $\perp$ となっている配列, $\mathcal{V}$ は初期状態として全て 0 となっている配列である. そして, $\mathsf{Ex}(z, \alpha)$ の出力は, α を用いて $z = \mathsf{H}^{\mathsf{P}}(m)$ となる m が見つかれば m を出力し, そうでなければ $\perp$ を出力する. そして, H^{P}, $\mathcal{A}$, $\mathcal{E}$ に対する PrA アドバンテージを以下に定義する.

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}} = \Pr[\mathsf{Exp}^{\mathsf{pra}}_{F^P,P,\mathcal{E},A} \Rightarrow \mathsf{true}]$$

そして, 任意の $\mathcal{A}$ に対して上記のアドバンテージが無視できるくらい小さい確率で抑えられるエクストラクタ $\mathcal{E}$ が存在する場合, H^{P} は PrA を満たすとする.

ところで, Dodis らによって, PrA のアドバンテージは衝突困難性のアドバンテージと 1-WPrA (1-Weak PrA) アドバンテージで得られることが示されている [27].

まず, 1-WPrA から説明する. この安全性では以下のゲームを考える.

$\underline{\mathsf{Exp}^{\mathsf{1wpra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}^+,\mathcal{A}}}$	$\underline{\textbf{oracle } \mathsf{h}(x)}$	$\underline{\textbf{oracle } \mathsf{Ex}^+(z)}$
$m^* \leftarrow \mathcal{A}^{\mathsf{h},\mathsf{Ex}^+}$;	$y \leftarrow \mathsf{P}(x)$;	$\mathsf{Q}[z] \leftarrow 1$;
$z \leftarrow \mathsf{H}^{\mathsf{P}}(m^*)$;	$\alpha \xleftarrow{\cup} (x, y)$;	$L \leftarrow \mathcal{E}^+(z, \alpha)$;
Ret $(x \notin L \wedge \mathsf{Q}[z] = 1)$;	Ret y;	Ret L;

PrA のゲームとの違いはエクストラクションオラクルとそのクエリ回数が違う. 1-WPrA では, マルチポイントエクストラクションオラクル E^+ を考える. そして, $\mathcal{A}$ から E^+ へのクエリ回数は 1 回である. また, E^+ は初期値が全て 0 となっている配列 Q と初期状態が $\emptyset$ となっているリスト L を持ち, マルチポイントエクストラクタ $\mathcal{E}^+$ のインターフェースを提供する. $\mathcal{E}^+(z, \alpha)$ は α から $z = \mathsf{H}^{\mathsf{P}}(m)$ となる全ての m の集合を出力する. そのような m が存在しない場合, $\emptyset$ を出力する. そして, 1-WPrA では, ハッシュ関数 H^{P}, 攻撃者 $\mathcal{A}$, マルチポイントエクストラクタ $\mathcal{E}^+$ に対して, 以下のアドバンテージを考える.

$$\mathsf{Adv}^{\mathsf{1wpra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}} = \Pr[\mathsf{Exp}^{\mathsf{1wpra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}^+,\mathcal{A}} \Rightarrow \mathsf{true}]$$

そして, 任意の攻撃者 $\mathcal{A}$ に対して, 上記のアドバンテージが無視できる確率で抑えられるマルチポイントエクストラクタ $\mathcal{E}^+$ が存在する場合, H^{P} を 1-WPrA と呼ぶことにする.

次に, ハッシュ関数 H^{P} と攻撃者 $\mathcal{A}$ に対する衝突困難性のアドバンテージを以下に定義する.

$$\mathsf{Adv}^{\mathsf{Coll}}_{\mathsf{H}^{\mathsf{P}}}(A) = \Pr[(m, m') \xleftarrow{\$} \mathcal{A}^{\mathsf{P}} : \mathsf{H}^{\mathsf{P}}(m) = \mathsf{H}^{\mathsf{P}}(m') \wedge m \neq m']$$

そして, PrA のアドバンテージは以下の補題の通り, 1-WPrA のアドバンテージと衝突困難性のアドバンテージから求めることができる.

補題 19 ([27] の補題 3.3 と補題 3.4) $\mathcal{E}^+$ を任意のマルチポイントエクストラクタとする. そして, q_e 回のエクストラクションクエリと q_P 回のプリミティブクエリを行う任意の PrA 攻撃者 $\mathcal{A}\mathsf{pra}$ に対して, 以下の式を満たす 1-WPrA 攻撃者 $\mathcal{A}^{\mathsf{1wpra}}$ と衝突困難性を破る攻撃者 $\mathcal{A}^{\mathsf{Coll}}$ が存在する.

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}}(\mathcal{A}^{\mathsf{pra}}) \leq q_e \cdot \mathsf{Adv}^{\mathsf{1wpra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}^+}(\mathcal{A}^{\mathsf{1wpra}}) + \mathsf{Adv}^{\mathsf{Coll}}_{\mathsf{H}^{\mathsf{P}}}(\mathcal{A}^{\mathsf{Coll}}).$$

ここで, $\mathcal{A}^{\mathsf{1wpra}}$ の動作時間は $\mathcal{O}(q_e\mathsf{Time}(\mathcal{E}^+))$, h へのクエリ回数は $\mathcal{A}^{\mathsf{pra}}$ と同じである. そして, $\mathcal{A}^{\mathsf{Coll}}$ の動作時間は $\mathcal{O}(q_e \cdot \mathsf{Time}(\mathcal{E}^+))$ で, P へのクエリ回数は q_P 回である. また, $\mathcal{E}$ の動作時間は $\mathcal{E}^+$ と同じである. ♦

PrA 安全なハッシュ関数から IFRO 安全なハッシュ関数の構成.

Dodis らは PrA の定義を用いて, H^{P} が PrA ならば H^{P} にランダムオラクル g を追加した $g \circ \mathsf{H}^{\mathsf{P}}$ は IFRO 安全性を満たすことを証明した [26, 27]. 具体的には以下のことを証明した.

補題 20 ([27] の定理 4.1) P を理想的な部品として, $\mathsf{H}^{\mathsf{P}} : \{0,1\}^* \to \{0,1\}^{2n}$ を P を用いたハッシュ関数とする. そして, $g : \{0,1\}^{2n} \to \{0,1\}^{2n}$ をランダムオラクルとする. そして, $\mathcal{E}$ を H^{P} に対する任意のエクストラクタとする. この場合, 任意の識別者 $\mathcal{D}$ に対して, 以下の式を満たす攻撃者 $\mathcal{B}$ とシミュレータ S が存在する.

$$\mathsf{Adv}^{\mathsf{indiff}}_{g\circ\mathsf{H}^{\mathsf{P}},\mathcal{RO},S}(\mathcal{D}) \leq \mathsf{Adv}^{\mathsf{pra}}_{\mathsf{H}^{\mathsf{P}},\mathcal{E}}(\mathcal{B}).$$

ここで, $S = (S_{\mathsf{P}}, S_g)$ で, S_{P} は P をシミュレートして, S_g は g をシミュレートする. $\mathcal{D}$ は 3 つのオラクル (L, R_{P}, R_g) にアクセスして, Real World である $(L, R_{\mathsf{P}}, R_g) = (g \circ \mathsf{H}^{\mathsf{P}}, \mathsf{P}, g)$ と Ideal World である $(L, R_{\mathsf{P}}, R_g) = (\mathcal{RO}, S_{\mathsf{P}}, S_g)$ を識別する. そして, $\mathcal{D}$ の (L, R_{P}, R_g) へのクエリ回数をそれぞれ q_L, q_{P}, q_g とする. l を 1 回の L へのクエリの最大ブロック長とする. H^{P} を 1 回読んだ時の最大時間を t_H, その時の P へのクエリ回数を q_H とする. S の実行時間は $\mathcal{O}(q_P + q_g \cdot \mathsf{Time}(\mathcal{E}))$ となる. また, $\mathcal{B}$ の実行時間は $\mathcal{O}(\mathsf{Time}(\mathcal{A}) + q_L t_H + q_{\mathsf{P}} + q_g)$ となる. ♦

SMD 構造による PrA 性の保存

Dodis らは SMD 構造を用いると, 圧縮関数 h が PrA ならば SMD^h も PrA となることを示した. 具体的には, 以下の通りである.

補題 21 ([27] の定理 4.2) P を理想的な部品として, h^P を P を用いた圧縮関数とする. そして, $\mathcal{E}_h$ を h^P に対する任意のエクストラクタとする. この場合, 任意の攻撃者 $\mathcal{A}_H$ に対して, 以下の式を満たす SMD^{h^P} に対するエクストラクタ $\mathcal{E}_H$ が存在する.

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{SMD}^{h^P},\mathcal{E}_H}(A_H) \leq \mathsf{Adv}^{\mathsf{pra}}_{h^P,\mathcal{E}_h}(A_h)$$

ここで, $\mathcal{A}_H$ の P へのクエリ回数を q_P 回, エクストラクタへのクエリ回数を q_e 回とする. また, l を $\mathcal{E}_H$ が出力するメッセージの最大ブロック長, q_H を SMD^{h^P} を 1 回計算する際の P の

最大呼び出し回数とする．この場合, $\mathcal{E}_H$ の動作時間は $l(\mathsf{Time}(\mathcal{E}_h) + \mathsf{Time}(\mathsf{unpad}))$, A_h の動作時間は $\mathcal{O}(\mathsf{Time}(A_H) + q_e l)$, P へのクエリ回数は $q_H + q_P$, エクストラクタへのクエリ回数は $q_e l$ となる．♦

6.3 IFRO 安全なブロック暗号ベースの倍ブロック長構造

6.3.1 提案倍ブロック長構造

$\mathsf{BC}_{2n,n} = (E, D)$ を鍵長が $2n$ bit, ブロック長が n bit のブロック暗号とする．以下, E を用いた Hirose の構造をベースとする倍ブロック長構造 F^E を定義する．

- 入力：$m \in \{0,1\}^*$
- 出力：$z \in \{0,1\}^{2n}$
- 以下の方法で, $z = F^E(m)$ を計算する．

$$z = f^E(\mathsf{SMD}^{\mathsf{Hirose}^E}(m))$$

ここで, SMD は Strengthened MD 構造で, f^E の構造は以下の通りである．

- 入力：$x \in \{0,1\}^{2n}$
- 出力：$y \in \{0,1\}^{2n}$
- c_1, c_2 を n bit の固定値として, 以下の方法で, $y = f^E(x)$ を計算する．

$$y = E(x, c_1) \| E(x, c_2)$$

6.3.2 提案倍ブロック長構造の安全性

提案倍ブロック長構造の安全性は以下の通りである．提案構造は理想的なブロック暗号を組み込んだ時に n bit の IFRO 安全性を満たす．以下, 理想的なブロック暗号について議論するときは添え字に I をつける．例えば, (E_I, D_I) など．

定理 6.1 $\mathcal{C}_{2n,n} = (E_I, D_I)$ を理想的なブロック暗号とする．この時, 任意の識別者 $\mathcal{D}$ に対して以下の式を満たすシミュレータ $S = (S_E, S_D)$ が存在する．

$$\mathsf{Adv}^{\mathsf{indiff}}_{F^{E_I},\mathcal{RO},S}(\mathcal{D}) \leq \frac{2Q^2}{(2^n - 2Q)^2} + \frac{2Q}{2^n - 2Q} + \frac{2l(2q)Q}{(2^n - Q)^2} + \frac{q_L + 2q}{2^n} + \frac{14Q}{2^n - Q}$$

ここで, $S = (S_E, S_D)$ であり, S_E は E_I をシミュレートして, S_D は D_I をシミュレートする. そして, $\mathcal{D}$ は 3 つのオラクル (L, R_E, R_D) にアクセスして, Real World である $(L, R_E, R_D) = (F^{E_I}, E_I, D_I)$ と, Ideal World である $(L, R_E, R_D) = (\mathcal{RO}, S_E, S_D)$ を識別しようとする. $\mathcal{D}$ の (L, R_E, R_D) へのクエリ回数をそれぞれ q_L, q_E, q_D 回として, L への 1 回のクエリの最大ブロック長を l とする. そして, $q = q_E + q_D$, $Q = 2l(q_H + 1) + q_E + q_D$ とする. この時, シミュレータの動作時間は $\mathcal{O}(q + 2lqQ) + 2lq \times \mathsf{Time}(\mathsf{unpad})$ で, クエリ回数は $2q$ となる. ♦

6.3.3 定理 6.1 の証明

以下の方針で定理 6.1 を証明する.

- **Step 1.** まず, 入出力長が $2n$ bit の関数 g と, 鍵長が $2n$ bit, ブロック長が n bit のブロック暗号 $\mathsf{BC}^1_{2n,n} = (E1, D1)$ を用いた以下の構造 F_1 を考える. ここで, $E1$ を暗号化関数, $D1$ を復号関数とする.
  - 入力 : $m \in \{0,1\}^*$
  - 出力 : $z \in \{0,1\}^{2n}$
  - 以下の方法で $z = F_1^{g,E1}(m)$ を計算する.

    $$z = g(\mathsf{SMD}^{\mathsf{Hirose}^{E1}}(m))$$

  そして, g がランダムオラクルで, $\mathsf{BC}^1_{2n,n}$ が理想的なブロック暗号の場合, F_1 が n bit の IFRO 安全性を満たすことを証明する.
- **Step 2.** 次に, $\mathsf{BC}^1_{2n,n}$ とは独立のブロック暗号 $\mathsf{BC}^2_{2n,n} = (E2, D2)$ を用いた入出力長が $2n$ bit の関数 f を考える. $f^{E2}(x) := E2(x, c_1) \| E2(x, c_2)$ である. そして, $\mathsf{BC}^2_{2n,n}$ が理想的なブロック暗号の場合, f が n bit の IFRO 安全性を満たすことを証明する. この結果と, Step 1 の結果を合わせると, $\mathsf{BC}^1_{2n,n}, \mathsf{BC}^2_{2n,n}$ が理想的なブロック暗号の場合以下の構造を持つ F_2 は n bit の IFRO 安全性を満たすことが保障できる.

  $$F_2^{E1,E2}(m) = f^{E2}(\mathsf{SMD}^{\mathsf{Hirose}^{E1}}(m))$$

- **Step 3.** 最後に, $\mathsf{BC}_{2n,n}, \mathsf{BC}^1_{2n,n}, \mathsf{BC}^2_{2n,n}$ が理想的なブロック暗号の場合, F^E はクエリ回数が 2^n まで F_2 と Indifferentiable となることを証明する. この結果と Step 2 の結果を合わせると, $\mathsf{BC}_{2n,n}$ が理想的なブロック暗号の場合, F^E が n bit の IFRO 安全性を満たすことが保障できる.

以下, それぞれのステップごとに証明を行っていく.

Step 1.

Step 1 では, $F_1^{g,E1}$ の IFRO 安全性を Dodis らの Preimage Awareness (PrA) [26, 27] を用いて証明する. Dodis らは圧縮関数 h が PrA を満たすならば SMD^h も PrA の性質を満たすことを証明した. そして, H^{P} が PrA ならばランダムオラクル g を用いた $g \circ \mathsf{H}^{\mathsf{P}}$ は IFRO 安全となることを証明した. すなわち, $\mathsf{BC}^1_{2n,n}$ が理想的なブロック暗号の場合, Hirose^{E1} が PrA となることを証明すれば, Step 1 の証明が完了する.

まず, Hirose^{E1} が PrA 安全性を満たすことを証明する.

補題 22 (Hirose の構造は PrA である) $\mathcal{C}^1_{2n,n} = (E1_I, D1_I)$ を理想的なブロック暗号とする. この時, 任意の攻撃者 $\mathcal{A}$ に対して, 以下の式を満たすエクストラクタ $\mathcal{E}$ が存在する.

$$\mathsf{Adv}^{\mathsf{pra}}_{\mathsf{Hirose}^{E1_I},\mathcal{E}}(\mathcal{A}) \leq \frac{2q_P^2}{(2^n - 2q_P)^2} + \frac{2q_P}{2^n - 2q_P} + \frac{2q_P q_e}{(2^n - q_P)^2}$$

ここで, $\mathcal{A}$ は $\mathcal{C}^1_{2n,n}$ に q_P 回クエリして, $\mathcal{E}$ に q_e 回クエリする. この時の $\mathcal{E}$ の実行時間は $\mathcal{O}(q_e q_P)$ となる. ♦

証明. まず, Hirose の構造が 1-WPrA となることを証明する. マルチポイントエクストラクタ $\mathcal{E}^+$ を以下に定義する.

algorithm $\mathcal{E}^+(z, \alpha)$
L を空列のテーブルとする;
α の要素を $(k_1, x_1, y_1), \ldots, (k_i, x_i, y_i)$ とする; $//E1_I(k_j, x_j) = y_j$ である
For $j = 1$ to i do
  If $z[1, n] = x_j \oplus y_j$ then
    $y \leftarrow E1_I(k_j, x_j \oplus C)$;
    If $z[n+1, 2n] = C \oplus x_j \oplus y$ then $L \xleftarrow{\cup} (x_j || k[1, n], k[n+1, 2n])$;
  If $z[n+1, 2n] = x_j \oplus y_j$ then
    $y \leftarrow E1_I(k_j, x_j \oplus C)$;
    If $z[1, n] = C \oplus x_j \oplus y$ then $L \xleftarrow{\cup} ((x_j \oplus C) || k[1, n], k[n+1, 2n])$;
L を返信する;

上記の手続きより, 1 ブロック目の $E1_I$ の入出力が定義された場合, 同時に 2 ブロック目の $E1_I$ の入出力も定義される. 同様に, 2 ブロック目の $E1_I$ の入出力が定義された場合, 同時に 1

ブロック目の $E1_I$ の入出力も定義される. よって, 上記の $\mathcal{E}^+$ と $\mathsf{Q}[z] = 1$ となる z に対して, $\mathcal{A}$ は z を計算する際の 1 ブロック目と 2 ブロック目の入出力を知らずに $z = \mathsf{Hirose}^{E1_I}(k, x)$ となる (k, x) を見つける必要がある. この解析は文献 [29] の一方向性の解析が適用できて, 一方向性は $2q_P/(2^n - q_P)^2$ で抑えられることが証明されている.

そして, 文献 [29] で Hirose の構造の衝突困難性のアドバンテージが $2q_P^2/(2^n - 2q_P)^2 + 2q_P/(2^n - 2q_P)$ となることが証明されているため, 補題 19 より上記の補題のバウンドを得ることができる.

■

そして, 上記の補題 22 を補題 20 と補題 21 と合わせることで以下の定理を得ることができる.

補題 23 任意の識別者 $\mathcal{D}$ に対して, 以下の式を満たすシミュレータ S_1 が存在する.

$$\mathsf{Adv}^{\mathsf{indiff}}_{F_1^{g,E1_I},\mathcal{RO},S1}(\mathcal{D}) \leq \frac{2Q_1^2}{(2^n - 2Q_1)^2} + \frac{2Q_1}{2^n - 2Q_1} + \frac{2lq_gQ_1}{(2^n - Q_1)^2}$$

ここで, $S_1 = (S1_g, S1_{\mathcal{C}})$, $S1_{\mathcal{C}} = (S1_E, S1_D)$ とする. $S1_g$ は g をシミュレートして, $S1_E$ は $E1_I$ をシミュレートして, $S1_D$ は $D1_I$ をシミュレートする. そして, $\mathcal{D}$ は 4 つのオラクル $(L, R_g, R_E.R_D)$ にアクセスして, Real World である $(L, R_g, R_E.R_D) = (F_1, g, E1_I, D1_I)$ のケースと Ideal World である $(L, R_g, R_E.R_D) = (\mathcal{RO}, S_g, S_E, S_D)$ のケースを識別する. l を L への 1 回のクエリの最大ブロック数とする. $\mathcal{D}$ から $(L, R_g, R_E.R_D)$ へのクエリ回数をそれぞれ q_L, q_g, q_E, q_D 回として, $Q_1 = 2l(q_H + 1) + q_E + q_D$ とする. S_1 の動作時間は $\mathcal{O}(q_E + q_D + lq_gQ_1) + lq_g \times \mathsf{Time}(\mathsf{unpad})$ で, $S1_g$ は $\mathcal{RO}$ に 1 回クエリする. ♦

Step 2.

まず, $\mathsf{BC}^2_{2n,n} = (E2, D2)$ が理想的なブロック暗号の場合, f^{E2} が n bit の IFRO 安全性を満たすことを示す.

補題 24 (f の IFRO 安全性) $\mathcal{C}^2_{2n,n} = (E2_I, D2_I)$ を理想的なブロック暗号とする. この場合, 任意の識別者 $\mathcal{D}$ に対して以下の式を満たすシミュレータ $S = (S_E, S_D)$ が存在する.

$$\mathsf{Adv}^{\mathsf{indiff}}_{f^{E2_I},\mathcal{RO},S}(\mathcal{D}) \leq \frac{q_f + q_E + q_D}{2^n}$$

simulator $\mathsf{S}_E(k,x)$

1. If $\mathsf{E}[k,x] \neq \perp$ then ret $\mathsf{E}[k,x]$;
2. If $\mathsf{E}[k,c_1] = \perp$ then
3. $\quad y^* \leftarrow \mathcal{RO}(k)$;
4. $\quad \mathsf{E}[k,c_1] \leftarrow y^*[1,n]$;
5. $\quad \mathsf{D}[k,\mathsf{E}[k,c_1]] \leftarrow c_1$;
6. $\quad \mathsf{E}[k,c_2] \leftarrow y^*[n+1,2n]$;
7. $\quad \mathsf{D}[k,\mathsf{E}[k,c_2]] \leftarrow c_2$;
8. If $x \neq c_1$ and $x \neq c_2$ then
9. $\quad \mathsf{E}[k,x] \xleftarrow{\$} \{0,1\}^n \backslash \mathcal{T}_E[k]$;
10. $\quad \mathsf{D}[k,\mathsf{E}[k,x]] \leftarrow x$;
11. Return $\mathsf{E}[k,x]$;

simulator $\mathsf{S}_D(k,y)$

1. If $\mathsf{D}[k,y] \neq \perp$ then ret $\mathsf{D}[k,y]$;
2. If $\mathsf{E}[k,c_1] = \perp$ then
3. $\quad y^* \leftarrow \mathcal{RO}(k)$;
4. $\quad \mathsf{E}[k,c_1] \leftarrow y^*[1,n]$;
5. $\quad \mathsf{D}[k,\mathsf{E}[k,c_1]] \leftarrow c_1$;
6. $\quad \mathsf{E}[k,c_2] \leftarrow y^*[n+1,2n]$;
7. $\quad \mathsf{D}[k,\mathsf{E}[k,c_2]] \leftarrow c_2$;
8. If $\mathsf{D}[k,y] \neq \perp$ then
9. $\quad \mathsf{D}[k,y] \xleftarrow{\$} \{0,1\}^n \backslash \{\mathcal{T}_D[k]\}$;
10. $\quad \mathsf{E}[k,\mathsf{D}[k,y]] \leftarrow y$;
11. Return $\mathsf{D}[k,y]$;

図.6.1　シミュレータ

ここで, $\mathcal{D}$ は 3 つのオラクル (L,R_E,R_D) にアクセスして, Real World である $(L,R_E,R_D) = (f^{E2_I}, E2_I, D2_I)$ と Ideal World である $(L,R_E,R_D) = (\mathcal{RO}, S_E, S_D)$ を識別しようとする. S_E は $E2_I$ をシミュレートして, S_D は $D2_I$ をシミュレートする. $\mathcal{D}$ の (L,R_E,R_D) へのクエリ回数をそれぞれ q_f, q_E, q_D とすると, S の動作時間は $\mathcal{O}(q_E+q_D)$, クエリ回数は q_E+q_D となる. ♦

証明. まず, シミュレータ $S=(S_E,S_D)$ を図 6.1 に定義する. このシミュレータは初期状態として全ての要素に $\perp$ が挿入されている配列 E,D と空列のテーブル $\mathcal{T}_E,\mathcal{T}_D$ を持つ.

この証明では以下の 3 つのゲームを考えて証明する. それぞれのゲームで $\mathcal{D}$ は 3 つのオラクル (L,R_E,R_D) と対話する.

- Game 0: このゲームは Real World である. すなわち, $(L,R_E,R_D) = (f^{E2_I}, E2_I, D2_I)$ となる.
- Game 1: このゲームでは $(L,R_E,R_D) = (f^{S_E}, S_E, S_D)$ とする.
- Game 2: このゲームは Ideal World である. すなわち, $(L,R_E,R_D) = (\mathcal{RO}, S_E, S_D)$ となる.

ここで, Game i で $\mathcal{D}$ が 1 を出力するイベントを G_i と書くと, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{f^{E3_I},\mathcal{RO},S}(\mathcal{D}) \leq \Pr[G_0] - \Pr[G_1] + \Pr[G_1] - \Pr[G_2]$$

以下, $\Pr[G_0] - \Pr[G_1]$ と $\Pr[G_1] - \Pr[G_2]$ を評価する.

$\Pr[G_0] - \Pr[G_1]$ の評価: Game 0 と Game 1 の差は (R_E, R_D) である. ここで, Game 0 では $(R_E, R_D) = (E2_I, D2_I)$, Game 1 では $(R_E, R_D) = (S_E, S_D)$ となる. そして, S_E と S_D は $\mathsf{E}[k, c_1] \neq \perp$ かつ $\mathsf{E}[k, c_2] \neq \perp$ となっている全ての k に対して, $\mathsf{E}[k, c_1] \neq \mathsf{E}[k, c_2]$ ならばシミュレータの手続きはレイジーサンプル手法による理想的なブロック暗号そのものの手続きを実現している. よって, $\Pr[G_0] - \Pr[G_1]$ はある k に対して $\mathsf{E}[k, c_1] = \mathsf{E}[k, c_2]$ となる確率で抑えられる. $\mathsf{E}[k, c_1]$ と $\mathsf{E}[k, c_2]$ の要素が定義される k の個数は $q_f + q_E + q_D$ 個なので, 以下の式が成り立つ.

$$\Pr[G_0] - \Pr[G_1] \leq \frac{q_f + q_E + q_D}{2^n}.$$

$\Pr[G_1] - \Pr[G_2]$ の評価: Game 1 と Game 2 の差は L で, Game 1 では $L = f^{S_E}$, Game 2 では $L = \mathcal{RO}$ である. Game 1 で L から S_E にクエリされているが Game 2 ではこのクエリは無い. そして, L の出力の計算方法も違うため, 以下の 2 点を証明する必要がある.

1. Game 1 で L から R_E へのクエリによって, $\mathcal{D}$ から L へのクエリ $(k, c_1), (k, c_2)$ に対するレスポンス y_1, y_2 が $y_1 \| y_2 = L(k)$ となっているため, Game 2 で $\mathcal{D}$ から L へのクエリ $(k, c_1), (k, c_2)$ に対するレスポンス y_1, y_2 が $y_1 \| y_2 = L(k)$ を満たすこと.
2. Game 1 で任意の L へのクエリ x に対して, $L(x) = \mathcal{RO}(x)$ となること.

ここで, シミュレータの定義から $\mathcal{D}$ から L へのクエリ $(k, c_1), (k, c_2)$ に対するレスポンス y_1, y_2 が $y_1 \| y_2 = \mathcal{RO}(k)$ となるため, 上記の 2 点は確率 1 で成り立つ. よって, $\Pr[G_1] - \Pr[G_2] = 0$ となる.

■

補題 24 で, $\mathsf{BC}^2_{2n,n}$ が理想的なブロック暗号の場合, f^{E2} が n bit の IFRO 安全性を満たすことを証明した. そして, 補題 23 により, F_1 が n bit の IFRO 安全性を満たすため, F_1 の関数 g を f に変更した F_2 も補題 23 と補題 24 から n bit の IFRO 安全性を満たす. 具体的には以下の定理が成り立つ.

補題 25 (F_2 の IFRO 安全性) $\mathcal{C}^1_{2n,n} = (E1_I, D1_I)$, $\mathcal{C}^2_{2n,n} = (E2_I, D2_I)$ を理想的なブロック暗号とする. この時, 任意の識別者 $\mathcal{D}$ に対して, 以下の式を満たすシミュレータ S が存在

する.

$$\mathsf{Adv}^{\mathsf{indiff}}_{F_2^{E1_I,E2_I},\mathcal{RO},S}(\mathcal{D}) \leq \frac{2Q^2}{(2^n-2Q)^2} + \frac{2Q}{2^n-2Q} + \frac{2lq_2Q}{(2^n-Q)^2} + \frac{q_L+q_2}{2^n}$$

ここで, $S = (S1, S2)$ として, $S1 = (S1_E, S1_D)$, $S2 = (S2_E, S2_D)$ とする. $(S1_E, S1_D)$ は $(E1_I, D1_I)$ をシミュレートし, $(S2_E, S2_D)$ は $(E2_I, D2_I)$ をシミュレートする. そして, $\mathcal{D}$ は 5 つのオラクル $(L, R_{E1}, R_{D1}, R_{E2}, R_{D2})$ にアクセスして, Real World である $(L, R_{E1}, R_{D1}, R_{E2}, R_{D2}) = (F_2, E1_I, D1_I, E2_I, D2_I)$ のケースと Ideal World である $(L, R_{E1}, R_{D1}, R_{E2}, R_{D2}) = (\mathcal{RO}, S1_E, S1_D, S2_E, S2_D)$ を識別しようとする. $\mathcal{D}$ から $(L, R_{E1}, R_{D1}, R_{E2}, R_{D2})$ へのクエリ回数はそれぞれ $(q_L, q_{E1}, q_{D1}, q_{E2}, q_{D2})$ とする. $\mathcal{D}$ から L への 1 回のクエリの最大ブロック長を l として, $Q = 2l(q_L+1)+q_{E2}+q_{D2}$, $q_2 = q_{E2}+q_{D2}$ とする. この時, S の動作時間は $\mathcal{O}(q_1 + lq_2Q) + lq_2 \times \mathsf{Time}(\mathsf{unpad})$, クエリ回数は q_2 となる. ♦

Step 3

本章では, $\mathsf{BC}_{2n,n} = (E, D)$, $\mathsf{BC}^1_{2n,n} = (E1, D1)$, $\mathsf{BC}^2_{2n,n} = (E2, D2)$ が理想的なブロック暗号の場合, F^E が $F_2^{E1,E2}$ と Indifferentiable であることを示す.

補題 26 $\mathcal{C}_{2n,n} = (E_I, D_I)$, $\mathcal{C}^1_{2n,n} = (E1_I, D1_I)$, $\mathcal{C}^2_{2n,n} = (E2_I, D2_I)$ を理想的なブロック暗号とする. この時, 任意の識別者 $\mathcal{D}$ に対して以下の式を満たすシミュレータ S が存在する.

$$\mathsf{Adv}^{\mathsf{indiff}}_{F^{E_I},F_2^{E1_I,E2_I},S}(\mathcal{D}) \leq \frac{14 \times (2(lq_F+1)+q_E+q_D)}{2^n-(2(lq_F+1)+q_E+q_D)}$$

ここで, $S = (S_E, S_D)$ として, S_E は E_I をシミュレート, S_D は D_I をシミュレートする. そして, $\mathcal{D}$ は 3 つのオラクル (L, R_E, R_D) にアクセスして, Real World である $(L, R_E, R_D) = (F^{E_I}, E_I, D_I)$ のケースと Ideal World である $(L, R_E, R_D) = (F_2^{E1_I,E2_I}, S_E, S_D)$ を識別しようとする. $\mathcal{D}$ から (L, R_E, R_D) へのクエリ回数はそれぞれ q_L, q_E, q_D として, l を 1 回の L へのクエリの最大ブロック数とする. この時, シミュレータの動作時間は $\mathcal{O}(3(q_E+q_D))$ で, クエリ回数は q_E+q_D である. ♦

証明. まず, この証明では F と F_2 のパディング関数を取り除いて議論する. すなわち, $\mathcal{D}$ から L へのクエリの長さは n の倍数ビットとなる. ここで, パディング関数を取り除いた F, F_2 の安全性はパディング関数を入れた F, F_2 の安全性をカバーすることに注意されたい.

simulator $\mathcal{S}_E(k,x)$

1. If $\mathsf{E}[k,x] \neq \perp$ then ret $\mathsf{E}[k,x]$
2. If $\mathsf{E}[k,c_1] = \perp$ then
3. $\quad y \leftarrow E2_I(k,c_1)$
4. $\quad \mathsf{E}[k,c_1] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_1$
5. $\quad y \leftarrow E2_I(k,c_2)$
6. $\quad \mathsf{E}[k,c_2] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_2$
7. If $x \neq c_1$ and $x \neq c_2$ then
8. $\quad y \leftarrow E1_I(k,x)$
9. $\quad \mathsf{E}[k,x] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow x$
10. Return $\mathsf{E}[k,x]$

simulator $\mathcal{S}_D(k,y)$

1. If $\mathsf{D}[k,y] \neq \perp$ then ret $\mathsf{D}[k,y]$
2. If $\mathsf{E}[k,c_1] = \perp$ then
3. $\quad y \leftarrow E2_I(k,c_1)$
4. $\quad \mathsf{E}[k,c_1] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_1$
5. $\quad y \leftarrow E2_I(k,c_2)$
6. $\quad \mathsf{E}[k,c_2] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow c_2$
7. If $\mathsf{D}[k,y] = \perp$ then
8. $\quad x \leftarrow D1_I(k,y)$
9. $\quad \mathsf{E}[k,x] \leftarrow y$; $\mathsf{D}[k,y] \leftarrow x$
10. Return x

図.6.2　Simulator

そして, 図 6.2 にシミュレータ $S = (S_E, S_D)$ を定義する. このシミュレータは 2 つの配列 E と D を持つ. これらの配列は初期状態として全て $\perp$ で初期化されている. このシミュレータは $\mathcal{C}^1_{2n,n}$ と $\mathcal{C}^2_{2n,n}$ を用いて Real World である $(L, R_E, R_D) = (F^E, E, D)$ の E, D をシミュレートする.

この証明では以下の 3 つのゲームを考えて証明する. それぞれのゲームで識別者 $\mathcal{D}$ は 3 つのオラクル (L, R_E, R_D) にアクセスする.

1. Game 1: このゲームは Real World である. すなわち, $(L, R_E, R_D) = (F^{E_I}, E_I, D_I)$ である.
2. Game 2: このゲームでは (R_E, R_D) を (E, D) から (S_E, S_D) に変更する. すなわち, $(L, R_E, R_D) = (F^{S_E}, S_E, S_D)$ となる.
3. Game 3: このゲームは Ideal World である. すなわち, $(L, R_E, R_D) = (F_2^{E1_I, E2_I}, S_E, S_D)$ である.

ここで, Game i で $\mathcal{D}$ が 1 を出力するイベントを G_i と書くことにすると以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{F^{E_I}, F_2^{E1_I, E2_I}, S}(\mathcal{D}) = \Pr[G_1] - \Pr[G_2] + \Pr[G_2] - \Pr[G_3].$$

そこで, 以下では $\Pr[G_1] - \Pr[G_2]$ と $\Pr[G_2] - \Pr[G_3]$ をそれぞれ評価する.

$\Pr[G_1] - \Pr[G_2]$ の評価: Game 1 と Game 2 の差は (R_E, R_D) で, Game 1 では $(R_E, R_D) =$

(E_I, D_I) で, Game 2 では $(R_E, R_D) = (S_E, S_D)$ である. ここで, S の定義から, $x \neq c_1$ かつ $x \neq c_2$ となる S_E への任意のクエリ (k, x) について $E1_I(k, x) \neq E2(k, c_1)$ かつ $E1_I(k, x) \neq E2(k, c_2)$ が成り立ち, かつ $y \neq E2_I(k, c_1)$ かつ $y \neq E2_I(k, c_2)$ となる S_D への任意のクエリ (k, y) について, $D1_I(k, y) \neq c_1$ かつ $D2_I(k, y) \neq c_2$ が成り立てば, (S_E, S_D) は理想的なブロック暗号として振る舞う. よって, $\Pr[G_1] - \Pr[G_2]$ は上記のコリジョンが起こる確率で抑えられるため, 以下の式が成り立つ.

$$\Pr[G_1] - \Pr[G_2]| \leq \frac{2Q}{2^n - Q}.$$

$\Pr[G_2] - \Pr[G_3]$ の評価: Game 2 と Game 3 の差は L で, Game 2 では F^{S_E}, Game 3 では $F^{E1_I, E2_I}$ となる. よって, Game 2 で任意の L へのクエリ m に対して, $F^{S_E}(m) = F^{E1_I, E2_I}(m)$ となれば Game 2 と Game 3 は同じゲームとして扱うことができる. すなわち, $\Pr[G_2] - \Pr[G_3]$ は Game 2 である L へのクエリ m に対して, $F^{S_E}(m) \neq F^{E1_I, E2_I}(m)$ となる確率で抑えられる. ここで, S_E の定義から, F から S_E への $x \neq c_1$ かつ $x \neq c_2$ となる任意のクエリ (k, x) について, $S_E(k, x) \oplus x \neq c_1$, $S_E(k, x) \oplus x \oplus C \neq c_1$, $S_E(k, x) \oplus x \neq c_2$, $S_E(k, x) \oplus x \oplus C \neq c_2$ ならば $F^{S_E}(m) = F^{E1_I, E2_I}(m)$ となる. すなわち, $\Pr[G_2] - \Pr[G_3]$ は Game 2 で上記のコリジョンが起こる確率で抑えられるため, 以下の式が成り立つ.

$$\Pr[G_2] - \Pr[G_3] \leq \frac{4Q}{2^n - Q}$$

以上より, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{F^{E_I}, F_2^{E1_I, E2_I}, S}(\mathcal{D}) \leq \frac{6Q}{2^n - Q}.$$

■

そして, 補題 25 と上記の補題 26 を組み合わせると定理 6.1 が得られる.

第 7 章

SHA-0 の衝突攻撃の改良

7.1 はじめに

ハッシュ関数は任意長の入力を固定長の値に圧縮する関数である. ハッシュ関数に求められる性質は主に以下の 3 つである. 以下, ハッシュ関数を $H : \{0,1\}^* \to \{0,1\}^n$ と書くことにする.

- 衝突困難性 : $H(m) = H(m')$ となる異なる 2 つの入力値 m, m' を $2^{n/2}$ より効率的な計算量で見つけることができない.
- 第 2 原像計算困難性 : m が与えられた時, $H(m) = H(m')$ となる異なる m とは異なる入力値 m' を 2^n より効率的な計算量で見つけることができない.
- 一方向性 : $z \in \{0,1\}^n$ が与えられた時, $z = H(m)$ となる m を 2^n より効率的な計算量で見つけることができない.

このような性質がハッシュ関数には求められるものの, 最近, ハッシュ関数の安全性解析が活発に議論されており, 重要なハッシュ関数である MD4, MD5, SHA-0, SHA-1 の衝突困難性を破る攻撃が発見された. EUROCRYPT2005 で, 中国の研究グループである Wang らによって MD5 の衝突困難性を 2^{39} の計算量で破る攻撃法を提案した [59, 61]. その後, CRYPTO 2005 で, Wang らは SHA-1 の衝突困難性を 2^{69} の計算量で破る攻撃法を提案した [60]. また, Wang らは, SHA-1 のベースとなるハッシュ関数である SHA-0 に対して, 従来提案されていた SHA-0 に対する衝突難性を破る攻撃は Biham らの 2^{51} で破る攻撃法であったが, SHA-1 と同様の手法を用いることで SHA-0 に対する衝突攻撃を 2^{39} の計算量に改良した [62].

7.1.1 Wang らの攻撃

Wang らが衝突攻撃を発表する前に用いられていた攻撃法は, 排他的論理和を用いた差分攻撃で, まず, ハッシュ関数に対して線形な差分パスを探索し, 高確率で成立する入力差分 ΔM を探索して, 次に総当たりでその差分パスを満たすメッセージ M を見つけて, $M' = M \oplus \Delta M$ として $H(M) = H(M')$ となる異なる 2 つのメッセージ M, M' を見つける攻撃法である.

Wang らは, 従来の手法に対して, 以下の 3 つのテクニックを導入して衝突攻撃法の改良手法を提案した.

1. 差分を排他的論理和を用いた線形差分から算術差分に変更
2. ハッシュ関数の上位ラウンドの差分パスを高い確率で成立するメッセージを探索するテクニックである message modification (MM) の提案

まず, 差分として算術差分を用いることで, MD4, MD5, SHA-0, SHA-1 の内部の演算の多くは算術演算で構成されるため, 従来の算術演算の部分の線形差分で生じる確率を全て 1 にすることができ, さらに, 線形差分では 1 つの差分の影響は 1 bit としていたが, 算術差分を用いることで, 桁上がりなどが扱えるようになり, 差分の影響を 1 bit だけではなく多くの bit に対して柔軟に扱えるようになる. そして, MD4, MD5, SHA-0, SHA-1 の上位ステップは入力メッセージがダイレクトに影響するステップなので, これらのステップのパスについては, パスを満たすようにメッセージを変更することで確率 1 で満たすことができることに注目した MM を提案して, 衝突攻撃を改良した.

Wang らは差分パスが成立するためのハッシュ関数の内部の値に対する条件を見つけて, この条件を満たすようにメッセージを探索すれば差分パスを満たすメッセージが見つかることを示した. この条件はハッシュ関数の内部の値のあるビットに注目した条件で, 差分パスに対して s 個条件があった場合, 差分パスを満たすメッセージを見つける計算量は 2^s となる. 以降では, この条件のことを衝突条件と呼ぶことにする.

message modification はいくつかの衝突条件を満たすメッセージを効率的に満たす方法であり, 残りの条件については全数探索で条件を満たすまでメッセージを探索する. Wang らは多くの衝突条件を効率的に満たすことができる message modification を提案して, MD4, MD5, SHA-0, SHA-1 の攻撃に成功した.

7.1.2 Wang らの攻撃の改良について

ここで, MM を適用できるステップを広げることができれば, 高い確率で満たすことができる衝突条件の数を増やすことができ, 結果として, Wang らの攻撃法の計算量を改良することができる.

本研究では, Wang らが提案した MM より多くのステップの衝突条件に適用できる新しい MM である Submarine Modification (SM) を提案して, SM を SHA-0 に適用して, SHA-0 の衝突攻撃の計算量を 2^{39} から 2^{36} に改良できることを示す.

7.1.3 既存の Message Modification と Submarine Modification

MM には basic MM (BMM) と advanced MM (AMM) の 2 つのタイプがある. ここで, これらの MM はハッシュ関数の内部のメッセージの使われ方によって使い分けられる. まずは, ハッシュ関数の内部の仕様を簡単に説明して, これらの MM の使い分け方について簡単に説明する.

MD4, MD5, SHA-0, SHA-1 は圧縮関数 h を繰り返す構造を持つ. 具体的には, 入力値 M をある値でパディングして, メッセージブロック $M_1, ..., M_l$ に対して, $i = 1 \ldots l$ について, $x_i = h(x_{i-1}, M_i)$ を計算し, x_l を出力する. ここで, x_i を chaninig value (CV) と呼ぶ. また, x_0 は固定値である.

そして, h は, ステップ関数が繰り返される構造を持ち, 各ステップでメッセージブロック M_i から生成される値を用いて CV を更新していく. 各ステップで用いられるメッセージブロック M_i から生成される値は, M_i を j 個のメッセージ $m_0, ..., m_{j-1}$ に分割して, 第 1 ステップから第 j ステップまでは $m_0, ..., m_{j-1}$ が順番に用いられて, $j+1$ ステップ以降のステップ関数で使われる値はメッセージ拡張関数を用いて $m_0, ..., m_{j-1}$ から生成された値を用いる. 以降, 1 から j ステップまでを bare ステップ, $j+1$ ステップ以降のステップを expanded ステップと呼ぶことにする. 例えば, MD4, MD5, SHA-0, SHA-1 の場合, $j = 16$ となる.

ここで, BMM は bare ステップに対する MM である. bare ステップでは入力値がそのままステップ関数の入力として使われるため, bare ステップの衝突条件に合わせて入力値を変更することで確率 1 で衝突条件を満たすメッセージを見つけることができる. この MM が BMM である. そして, AMM は expanded ステップの MM のことである. expanded ステップで使うメッセージは bare ステップで使われたメッセージをメッセージ拡張関数を用いて生成する. ここで, bare ステップで BMM を用いて衝突条件を満たすメッセージを生成しているため,

expanded ステップの衝突条件を満たすメッセージを見つけるためには, 偶然条件を満たすようにするか bare ステップの衝突条件を壊さないようにメッセージを変更して衝突条件を満たすようにするかである. このメッセージ変更をする方法が AMM である.

ここで, AMM には MD4 と MD5 で提案された MM である Cancel Modification (CM) と SHA-0 と SHA-1 で提案された MM である Propagation Modification (PM) がある.

Cancel Modification (CM): まず, CM について説明する. CM は MD4 と MD5 の衝突攻撃で提案された MM である [59, 61]. CM の手続きは次の 2 つのステップから構成される. 以下の説明はターゲットとなる衝突条件が満たされていない場合を想定して説明する.

1. h にターゲットとなる衝突条件が変更されるようにメッセージブロックに差分を加える. 本研究ではこの差分を direct 差分と呼ぶことにする.
2. 次に, direct 差分が他の衝突条件に影響を与えないようににするためにメッセージブロックに direct 差分の影響を打ち消す差分を入力して, ターゲットの衝突条件があるステップまで計算する.

上記のステップ 1 の操作から, 衝突条件に関するビットが変更されて, 衝突条件を満たすことができる. そして, ステップ 1 の差分は圧縮関数の構造から bare ステップのメッセージに現れるため, 他の衝突条件に影響しないように, この差分を打ち消す必要がある. そのため, ステップ (2) により打ち消すための差分をメッセージに入れてこの差分の影響を打ち消す.

ここで, CM は MD4 と MD5 の衝突攻撃で提案された MM であるが, 本研究で, SHA-0 と SHA-1 にも適用して SHA-0 と SHA-1 のステップ 19 までの衝突条件を確率 1 で満たすことができることを示す. この詳細は 7.4.1 章で述べる.

Propagation Modification (PM): PM は SHA-0 と SHA-1 のの衝突攻撃で提案された MM である. 文献 [62] では, PM は SHA-0 のステップ 20 までの衝突条件に適用されていた.

ここで, PM について説明する. PM の手続きは次の 2 ステップから構成される以下の説明ではターゲットとなる衝突条件が満たされていない場合を想定して説明する.

1. 衝突条件に影響を及ぼす差分を発生させるために bare ステップに差分を入力する. 本研究ではこの差分を source 差分と呼ぶことにして, source 差分が入力されるステップを appear ステップと呼ぶことにする.
2. ターゲットとなる衝突条件があるステップまで CV を計算する.

ここで, 文献 [62] では SHA-0 のステップ 20 までの衝突条件に対して PM が適用されてい

たが, 我々は計算機実験によりステップ 21 まで適用できることを確認した.

7.1.4 本研究の成果：Submarine Modification (SM) の提案

本研究では CM と PM を組み合わせた新しい MM である Submarine Modification (SM) を提案する.

まず, SM では, CM のアイディアを用いて appear ステップの範囲を拡張する. ここでは, CM の source 差分を direct 差分としてみなし, そして, PM のテクニックを用いて direct 差分をターゲットとなる衝突条件まで伝搬させて対象となる衝突条件を満たすようにする. そして, SHA-0 と SHA-1 に対して submarine modification を適用し, ステップ 24 まで確率ほぼ 1 で衝突条件を満たせ, ステップ 25 の衝突条件は確率 0.85 で満たすことができることを計算機実験により確認した. この詳細は 7.5.2 章で述べる.

そして, 本研究では SM を Wang らが提案した SHA-0 の衝突条件に適用して, SHA-0 の衝突攻撃の計算量を 2^{39} から 2^{36} に改良することができることを示す.

7.2 攻撃対象とする圧縮関数

本章で, いくつかのハッシュ関数の構造について説明する.

MD4, MD5, SHA-0, SHA-1 といった多くのハッシュ関数は Merkle-Damgård 構造を用いて設計されている. この構造は以下のように圧縮関数 $h : \{0,1\}^u \to \{0,1\}^v$ を繰り返す構造となっている: まず, ハッシュ関数の入力 M の後ろに 1 を付加し, 次に 0 のビット列を付加し, 最後に 64bit の $|M|$ の値を付加した値で $u-v$ の倍数の長さとなる M^* を生成する. 次に, M^* を $u-v$ bit 毎に分割した値を $M_0, ..., M_n$ として, x_0 を v bit の固定値として, $i = 1, ..., n$ に対して, $x_i = h(x_{i-1}, M_i)$ を計算する. そして, x_n をハッシュ関数の出力値とする.

次に, MD5, SHA-0, SHA-1 の圧縮関数 h の仕様を説明する.

MD5 の圧縮関数

MD5 のパラメータ u, v, i, n の値は $u = 640, v = 128, i = 15, n = 64$ である.

a_0, b_0, c_0, d_0 をそれぞれ 32 bit の chaining value (CV) とする. ここで, j 番目の圧縮関数では $(a_0, b_0, c_0, d_0) = x_{j-1}$ となる. そして, 512 bit のメッセージ M_j に対して, MD5 の圧縮関数では次のように計算される. まず, M_j は 32 bit のメッセージ $m_0, ..., m_{15}$ に分割される. そして, 32 bit の値 $m_{16}, ..., m_{63}$ はメッセージ拡張関数によって生成される. ここで, $m_0, ..., m_{15}$ は初めの 16 ステップで使われるメッセージで, このステップが bare ステップと

なり, 残りの $m_{16}, ..., m_{63}$ はステップ 17 から 64 で使われて, このステップが expanded ステップとなる.

次に, 圧縮関数内で CV を更新するステップ関数の仕様を説明する. ここで, $i_1 = i_2 = i_3 = i_4 = 0$ とする. そして, $l = 1, ..., 64$ に対して, 以下の計算を行う.

$$
\begin{aligned}
a_{i_1+1} &= b_{i_2} + ((a_{i_1} + f_l(b_{i_2}, c_{i_3}, d_{i_4}) + m_{l-1} + k_{l-1})), \\
&\qquad i_1 = i_1 + 1 \text{ if } l \bmod 4 = 1 \\
d_{i_4+1} &= a_{i_1} + ((d_{i_4} + f_l(a_{i_1}, b_{i_2}, c_{i_3}) + m_{l-1} + k_{l-1})), \\
&\qquad i_4 = i_4 + 1 \text{ if } l \bmod 4 = 2 \\
c_{i_3+1} &= d_{i_4} + ((c_{i_3} + f_l(d_{i_4}, a_{i_1}, b_{i_2}) + m_{l-1} + k_{l-1})), \\
&\qquad i_3 = i_3 + 1 \text{ if } l \bmod 4 = 3 \\
b_{i_2+1} &= c_{i_3} + ((b_{i_2} + f_l(c_{i_3}, d_{i_4}, a_{i_1}) + m_{l-1} + k_{l-1})), \\
&\qquad i_2 = i_2 + 1 \text{ if } l \bmod 4 = 0
\end{aligned}
$$

ここで, k_{l-1} は固定値である. そして, $a_0 + a_{16}, b_0 + b_{16}, c_0 + c_{16}, d_0 + d_{16}$ が MD5 の圧縮関数の出力値となる.

ここで, 関数 f の仕様は以下の通りである.

- $f_s(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ $1 \leq s \leq 16$
- $f_s(x, y, z) = (x \wedge z) \vee (y \wedge \neg z)$ $17 \leq s \leq 32$
- $f_s(x, y, z) = x \oplus y \oplus z$ $33 \leq s \leq 48$
- $f_s(x, y, z) = y \oplus (x \vee \neg z)$ $49 \leq s \leq 64$

SHA-0 と SHA-1 の圧縮関数

$u = 672, v = 160, i = 15, n = 80$ として, a_0, b_0, c_0, d_0, e_0 を 32 bit の圧縮関数の CV の初期値とする. ここで, j 番目の圧縮関数の呼び出しの場合, $(a_0, b_0, c_0, d_0, e_0) = x_{j-1}$ となる.

SHA-0 と SHA-1 の圧縮関数では, まず, 512 bit のメッセージ M_j が 32bit のメッセージ $m_0, ..., m_{15}$ に分けられる. 次に, メッセージ拡張関数を使って 32bit の値 $m_{16}, ..., m_{79}$ が生成される. ここで, メッセージ拡張関数の仕様は以下の通りである. $16 \leq j \leq 79$ に対して,

- $m_j = m_{j-3} \oplus m_{j-8} \oplus m_{j-14} \oplus m_{j-16}$ (SHA-0)
- $m_j = (m_{j-3} \oplus m_{j-8} \oplus m_{j-14} \oplus m_{j-16}) \lll 1$ (SHA-1)

ここで, ステップ 1 から 16 が bare ステップ, ステップ 17 から 80 が expanded ステップとなる.

次に, $l = 1, \ldots, 80$ について, 以下を計算する.

$$\begin{aligned} a_l =& (a_{l-1} \lll 5) + f_l(b_{l-1}, c_{l-1}, d_{l-1}) \\ & + e_{l-1} + m_{l-1} + k_{l-1} \\ b_l =& a_{l-1} \\ c_l =& b_{l-1} \lll 30 \\ d_l =& c_{l-1} \\ e_l =& d_{l-1} \end{aligned}$$

ここで, k_{l-1} は 32 bit の固定値である. そして, $a_0 + a_{80}, b_0 + b_{80}, c_0 + c_{80}, d_0 + d_{80}, e_0 + e_{80}$ が圧縮関数の出力値となる.

関数 f の仕様は以下の通り.

- $f_l(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ $1 \leq l \leq 20$
- $f_l(x, y, z) = x \oplus y \oplus z$ $21 \leq l \leq 40$, $61 \leq l \leq 80$
- $f_l(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ $41 \leq l \leq 60$

7.3 Wang らの衝突攻撃

Wang らの衝突攻撃は算術演算を用いた差分攻撃法である. もしハッシュ関数 H に対して衝突が見つかるならば, すなわち, $H(M) = H(M'), M \neq M'$ となることなる 2 つのメッセージ M, M' が見つかるならば, 差分値は $\Delta M = M' - M \neq 0$, $\Delta H(M, M') = H(M') - H(M) = 0$ となる. ここで, ある値 x, x' に対して, $\Delta x := x' - x$ として, Δx を x の差分値と呼ぶことにする. 差分攻撃法では $\Delta M \neq 0$ なので, ハッシュ関数の CV の差分値は 0 ではない値となる.

Wang らの攻撃法はまず差分パスを見つけることからスタートする. この差分パスは, ハッシュ関数の出力差分値 $\Delta H(M, M')$ が $\Delta H(M, M') = 0$ となるように, CV の差分値と入力メッセージの差分値を決定する. SHA-0 や SHA-1 のこれらの差分値は local collision と呼ばれるテクニックを使って決められる. [62, 60] を参照.

しかし, $M - M' = \Delta M$ となるメッセージを使ったとしても, 圧縮関数内部の CV が満たしてほしい差分値となるとは限らないため, その出力差分が $H(M') - H(M) = 0$ となるとは限らない. 圧縮関数内部の CV の差分値が満たしてほしい差分値となるためには衝突条件と呼ばれる条件を満たす必要があり, 衝突条件を全て満たす場合, 出力差分が 0 となる. 衝突条件は M, M' を探索する前に事前に求めておく.

そして, 事前に求めておいた衝突条件とメッセージ差分 ΔM を用いてコリジョン探索を行う. コリジョン探索の第 1 ステップとして, 全ての衝突条件を満たすメッセージ M を見つけ

る. 次に $M' = M + \Delta M$ とする. ここで, M は全ての衝突条件を満たすメッセージとすると, M, M' は $H(M) = H(M')$ を満たす. ここで, 全ての衝突条件を満たす M を効率的に探索するために message modification (MM) が用いられる.

7.4 Message Modification (MM)

MM には basic MM (BMM) と advanced MM (AMM) がある.

本稿では, 主に AMM に注目する. そして, AMM には cancel modification (CM) [59, 61] と propagation modification (PM) [62] がある. 以下, これらの AMM について説明する.

7.4.1 Cancel Modification (CM)

CM は MD4 [59] や MD5[61] に対する衝突攻撃に対して提案された AMM である.

CM は, expanded ステップの衝突条件が満たされていない場合に適用する手法で, この衝突条件を満たすように適切な差分を入力する. 我々はこの差分を direct 差分と呼ぶことにする. もし, ステップ j の i 番目の bit の衝突条件が満たされていない場合, ステップ j のメッセージに差分を入力しこの衝突条件を満たすようにする. しかし, expanded ステップのメッセージは bare ステップのメッセージから計算されるため, この差分は bare ステップのメッセージにも現れる.

ここで, この差分は既に満たされている衝突条件に影響を与える可能性があるため, この差分の影響を消すために bare ステップのメッセージに新たな差分を入力する. CM は 7.1 章で述べた CM のステップ 1 と 2 から構成される. 図 7.1 は CM を図示したものである.

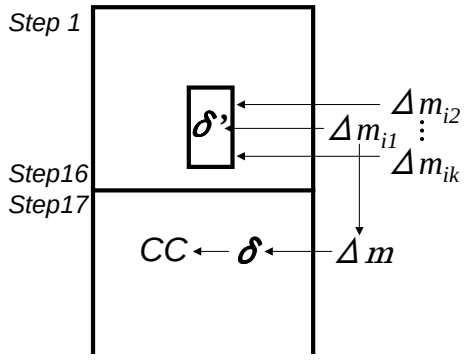

図.7.1 Cancel Modification

ここで, 例として, MD5 のステップ 17 の衝突条件 $a_{5,31} = 0$ に対する cM について説明す

る. $x_{i,j}$ はステップ i の値 x の j 番目の bit の値で, 31 番目の bit は MSB で 0 番目の bit は LSB であることに注意されたい. そして, 衝突条件が満たされていない場合, 以下の手続きを実行する.

1. $m_1 \leftarrow m_1 + 2^{26}$ and calculate a_5
2. $d_1 \leftarrow a_1 + (d_0 + f_1(a_1, b_0, c_0) + m_1 + t_1) \lll 5$
   $m_2 \leftarrow ((c_1 - d_1) \ggg 17) - c_0 - f_1(d_1, a_1, b_0) - t_2$
   $m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - f_1(c_1, d_1, a_1) - t_3$
   $m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - f_1(b_1, c_1, d_1) - t_4$
   $m_5 \leftarrow ((d_2 - a_2) \ggg 12) - d_1 - f_1(a_2, b_1, c_1) - t_5$

ここで, 上記のステップ 1 は 7.1 章で述べた CM のステップ 1 に対応するステップで, 上記のステップ 2 は 7.1 章で述べた CM のステップ 2 に対応するステップである. そして, a_5 は $a_5 = b_4 + (a_4 + f_2(b_4, c_4, d_4) + m_1 + t_{17}) \lll 5$ によって計算されることから, m_1 に差分 2^{26} を入力すると, $a_{5,31}$ に差分 2^{31} が表れて, MD5 の仕様から m_1 の差分 2^{26} は d_1 に伝搬する. そして, この差分はステップ 2 より m_2, m_3, m_4, m_5 の差分により打ち消される.

7.4.2 Propagation Modification (PM)

PM は SHA-0 の衝突攻撃で提案された AMM である [62]. PM は, ある衝突条件に対して, bare ステップのメッセージに適切な差分を入力して, 対象の衝突条件の bit に差分が現れるようにする手法である. 本稿では, このメッセージの差分を source 差分と呼び, 衝突条件がある bit に現れる差分を target 差分と呼ぶことにする. そして, source 差分が現れるステップを appear ステップと呼ぶことにする. この差分は source 差分を伝搬させることで得られるため, この伝搬した差分が target 差分となり衝突条件の bit を反転させることで, 衝突条件を満たすことができる. ここで, PM は 7.1 章で述べた PM の 2 つのステップから構成される. ここで, PM を図 7.2 に記載しておく.

ここで, SHA-0 のステップ 16 の衝突条件 $a_{17,31} = 0$ を例に PM を説明する. PM では, $a_{17,31} \neq 0$ の場合, 以下の手続きを実行する.

1. $m_{15} \leftarrow m_{15} \pm 2^{26}$.
2. ステップ 16 以降の CV を計算する.

ここで, 上記のステップ 1 は 7.1 章で述べた PM のステップ 1 に対応し, 上記のステップ 2 は 7.1 章で述べた PM のステップ 2 に対応する.

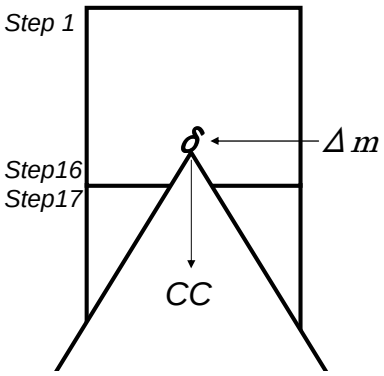

図.7.2　Propagation Modification

m_{15} の差分 2^{26} より, a_{16} に source 差分 2^{26} が現れて, この差分が a_{17} に伝搬して差分 2^{31} が現れる. この差分により, a_{17} の衝突条件を満たすことができる.

7.5　Submarine Modification の提案

本章で, CM と PM の 2 つのアイディアを用いた新しい AMM である submarine modification (SM) を提案する. ここでは主に SHA-0 と SHA-1 に適用することを考えて議論する.

PM の appear ステップは bare ステップであるが, SM では, appear ステップの範囲を CM を用いて拡大する. この範囲を広げるために, CM の direct 差分を PM の source 差分に適用する. Direct 差分は bare ステップ以降のステップに出現する差分であるため, これにより appear ステップの範囲を広げることができ, 結果として, AMM の適用範囲を広げることができる.

この SM を用いることで, ステップ 24 までの衝突条件を確率ほぼ 1 で満たすことができ, ステップ 25 の衝突条件を確率ほぼ 0.85 で満たすことを計算機実験により確認した.

7.5.1　Submarine Modification の構成法

ここで, SM の構成法を述べる.

1. ターゲットの衝突条件に対して PM のテクニックを用いて source 差分を決める. ここで, この差分は衝突条件を満たす成功確率が最も高くなるもで, 他のすでに満たされている衝突条件を変えないようなものを選ぶ. もし, 上記の source 差分が存在しない場合

は, ターゲットの衝突条件に対する MM は無しと判断する.

2. 次に, もし上記の手続きで source 差分が存在した場合, source 差分とは別の差分 (extra 差分と呼ぶ) と, extra 差分が成立するための条件 (extra 条件と呼ぶ) を選ぶ. ここで, extra 差分は, CM のアイディアを拡張した差分で, すでに成立している衝突条件を source 差分によって変えないようにするための差分である. Extra 条件は, extra 差分が成立するための条件で, CV とメッセージの bit に関する条件である. ここで, もし source 差分の影響を打ち消すことができる extra 差分が存在しなければ上記のステップ 1 に戻り, 他の source 差分を選びなおす.

7.5.2 SHA-0 と SHA-1 への適用

ここで, SM を SHA-0 と SHA-1 に適用した結果を説明する. SM には PM のアイディアを用いるステップと CMn のアイディアを用いるステップからなるため, 以下それぞれのステップについて順番に説明する.

7.5.2.1 Phase 1 (PM パート)

このフェーズでは, PM のアイディアを用いる. このテクニックを用いて, 高い確率で差分が伝搬して, 衝突条件を満たすことができるパタンをいくつか見つける.

ここで, このパタンには以下の 2 つのパタンに分けることができる. Pattern 1 の技法は, ステップ 24 までの衝突条件を確率ほぼ 1 で満たすことができ, さらにステップ 25 の衝突条件を確率およそ 0.85 で満たすことができる技法である. Pattern 2 の技法は, ステップ 23 までの衝突条件を確率ほぼ 1 で満たすことができ, さらにステップ 24 の衝突条件を確率およそ 0.85 で満たすことができる技法である.

ここで, 上記の確率は, ターゲットとなる衝突条件が満たされて, かつ他のすでに満たされている衝突条件が壊されない確率である. 我々はこのような確率を計算機実験によって導出した. 計算機実験では, メッセージ毎回ランダムに選び, 各々の衝突条件に対して 100000 回条件が成立するかチェックを行った. 具体的な確率は, 2 つのパタンの説明後に示す.

Pattern1

このパタンは 5bit 左巡回シフトを利用したパタンで, この 5bit 左巡回シフトにより差分も 5bit 左巡回シフトされることに注目する. ここで, 他の差分と離れた位置にこの差分を立てることで, 他の差分との影響を小さい確率に抑えることができる. CV が 32bit である点と 5bit

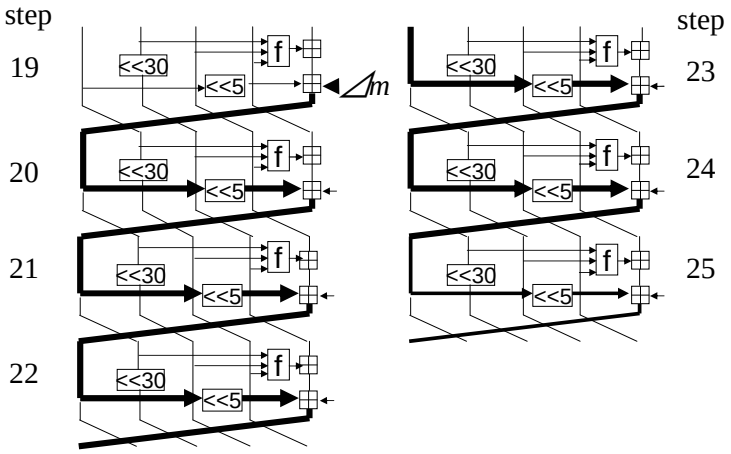

図.7.3　パタン 1

左巡回シフトを考慮に入れると, source 差分が出現可能なステップから 7 ステップ以降は高い確率で伝搬する差分が存在しない.

図 7.3 はステップ 19 に source 差分を入れた時の差分の伝搬を表した図で, ステップ 24 までの太線は確率ほぼ 1 で伝搬する差分パスを表し, ステップ 25 の太線は確率およそ 0.85 で伝搬する差分パスを表す. ここで, ステップ 26 以降は高い確率で伝搬する差分は存在しないことに注意されたい. このパタンでは, ステップ 19 で 2^j の source 差分が入力された時の図で, この差分により, a_{19} に差分 2^j が現れて, a_{20} に差分 2^{j+5}, a_{21} に差分 2^{j+10}, a_{22} に差分 2^{j+15}, a_{23} に差分 2^{j+20}, そして a_{24} に差分 2^{j+25} がそれぞれ 1 で現れる. そして, a_{25} に差分 2^{j+30} が確率およそ 0.85 で現れることが表現されている.

上記の差分を用いると, ターゲットとなる衝突条件に対して, 容易に source 差分の候補を選ぶことができる. 例えば, a_{23} の 25 番目の bit に関する条件に対する source 差分の候補はステップ 19 の 2^5 となる.

また, source 差分がステップ 17 やステップ 18 に入力された場合でも, 同様に差分パスの候補を求めることができる. 例えば, ステップ 18 に source 差分として 2^j を入れた場合, a_{18} に差分 2^j が出現し, それ以降は, a_{19} に差分 2^{j+5} が, a_{20} に差分 2^{j+10} が, a_{21} に差分 2^{j+15} が, a_{22} に差分 2^{j+20} が, a_{23} に差分 2^{j+25} がそれぞれ確率 1 で現れて, そして, a_{24} に差分 2^{j+30} が確率およそ 0.85 で現れる.

また, ステップ 17 に source 差分 2^j を入れた場合も同様に, a_{17} に差分 2^j が出現し, それ以降は, a_{18} に差分 2^{j+5} が, a_{19} に差分 2^{j+10} が, a_{20} に差分 2^{j+15} が, a_{21} に差分 2^{j+20} が, a_{22} に差分 2^{j+25} がそれぞれ確率 1 で現れて, そして, a_{23} に差分 2^{j+30} が確率およそ 0.85 で現

れる.

Pattern 2

図 7.4 を用いてこのパタンを説明する. ここの図では, source 差分がステップ 19 で入力され

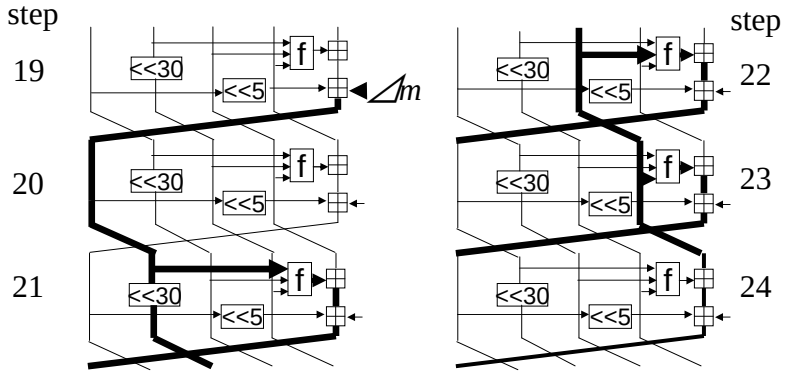

図.7.4　パタン 2

た時の差分の伝搬を表しており, 差分の伝搬を太線で表現している. このパタンでは, source 差分はステップ 23 まで確率ほぼ 1 で伝搬し, ステップ 24 では確率およそ 0.85 で伝搬する. 例えば, ステップ 19 で差分 2^j が入力された場合, 差分 2^j, 2^{j-2}, 2^{j-2} がそれぞれ a_{21}, a_{22}, a_{23} に確率 1 で伝搬し, そして, a_{24} に差分 2^{j-2} が確率およそ 0.85 で伝搬する.

このパタンを用いると, ある衝突条件に対していくつかの source 差分の候補を求めることができる. 例えば, a_{23} の 30 bit 目に衝突条件がある場合, ステップ 19 の差分 2^{28} が source 差分の候補となる.

また, ステップ 17 やステップ 18 に source 差分を入れても同様に議論することができる. ステップ 18 に source 差分 2^j を入れた場合, 2^{j-2}, 2^{j-2} が確率ほぼ 1 で a_{21}, a_{22} にそれぞれ伝搬し, 差分 2^{j-2} が確率ほぼ 0.85 で a_{23} に伝搬する. また, ステップ 17 の場合も同様に, ステップ 17 に source 差分 2^j を入れた場合, 差分 2^{j-2} が確率ほぼ 1 で a_{21} に伝搬し, 差分 2^{j-2} が確率ほぼ 0.85 で a_{22} に伝搬する.

ここで, 第 1 ラウンドの関数 f は IF 関数であることと第 2 ラウンドの関数 f は XOR 関数であることから, source 差分をステップ 18 に入力された場合, a_{20} に高い確率で伝搬する差分は存在しないことに注意されたい. すなわち, 第 1 ラウンドの関数 f の入力に差分が存在する場合, その出力差分が出現する確率は 1/2 であり, 第 2 ラウンドの関数 f の入力に差分が存在する場合, その出力差分は確率 1 で出現する. また, 同様の理由より, source 差分をステップ 17 に入れた時に, a_{19} と a_{20} に高い確率で成立する差分は存在しない.

SM の確率の導出方法

SM の成立確率を理論的に導出するのではなく, 計算機実験によって導出する.

ターゲットとなる衝突条件に対して, 上記で議論したパタン 1 とパタン 2 で得られる全てのパタンに対して, ターゲットとなる衝突条件が満たされる確率とすでに成立している衝突条件が壊される確率を 100000 回計算機実験を行うことで見積もった. この計算機実験では, 毎回ランダムにメッセージを選んでいる.

7.5.2.2 Phase 2 (CM パート)

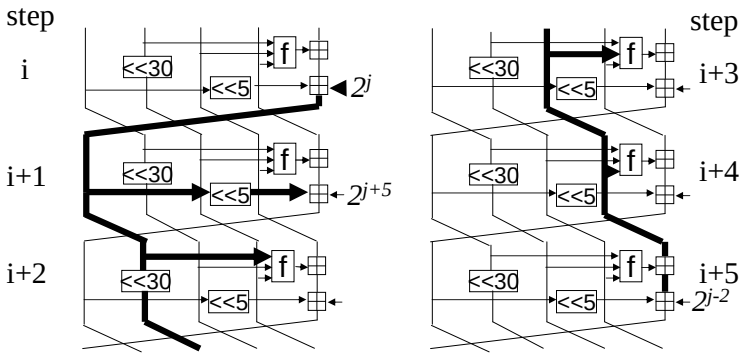

図.7.5 Bare ステップでの打ち消すための差分

SM のフェーズ 2 では, CM のテクニックを用いる. このテクニックを用いることで, いくつかの差分を打ち消すことができるパタンを見つけることができる. 以下, bare ステップの差分を打ち消す方法について説明する.

Cancellation of Differentials in the Bare Steps:

メッセージに差分が入力された場合, CV a にその差分が現れる. そして, この差分は差分が入力されたステップから 6 ステップ後まで高い確率で伝搬する. すなわち, 6 ステップより後のステップでは差分を打ち消すことができないため, 差分が入力されてから 6 ステップ分に注目し, この中で打ち消す方法を議論する.

図 7.5 はある差分を打ち消す方法を示している. ここでは, 主にフェーズ 2 が用いられていて, 関数 f の特徴とメッセージ差分を用いることで, 差分をキャンセルしている. また, 他のパタンでも図 7.5 と同様の方法で差分を打ち消すことができるものの, ステップ $i+2$ または $i+3$ または $i+4$ の f で差分を打ち消すことは困難であることに注意されたい. よって, ス

テップ $i+2$ の差分は関数 f の性質を利用するかメッセージ差分として 2^j を入れることでキャンセルすることができ, ステップ $i+3$ の差分は関数 f の性質を利用するかメッセージ差分として 2^{j-2} を入れることでキャンセルすることができ, ステップ $i+4$ の差分は関数 f の性質を利用するかメッセージ差分として 2^{j-2} を入れることでキャンセルすることができる. よって, 2^3 個のキャンセルできるパタンが存在する.

Extra 条件は, 衝突条件とは異なる条件で, 差分を打ち消すための差分を構成するために必要な条件である. ここで, 図 7.5 を用いて Extra 条件を説明する. この図では, Extra 条件として $a_{i,j} = m_{i-1,j}, m_{i,j+5} \neq m_{i-1,j}, a_{i-1,j+2} = a_{i-2,j+2}, a_{i+1,j-2} = 0, a_{i+2,j-2} = 1, m_{i+4,j-2} \neq m_{i-1,j}$ を考えている. この条件のうち $a_{i,j} = m_{i-1,j}$ は a_i に差分を立てないための条件, $m_{i,j+5} \neq m_{i-1,j}$ はステップ $i+1$ の差分をキャンセルするための条件, $a_{i-1,j+2} = a_{i-2,j+2}, a_{i+1,j-2} = 0, a_{i+2,j-2} = 1$ はステップ $i+2, i+3, i+4$ の関数 f で差分をキャンセルするための条件, $m_{i+4,j-2} \neq m_{i-1,j}$ はステップ $i+5$ で差分をキャンセルするための条件である.

Appear ステップの範囲:
文献 [62] では, appear ステップの範囲をステップ 16 までとしていたが, 本研究の成果により, この範囲を最大ステップ 19 まで拡張することができる.

7.5.3 SM の適用例

ここで, 文献 [62] と [44] に示されている, 差分パスに SM を適用する. そして, SHA-0 の攻撃計算量を 2^{39} 回の SHA-0 の圧縮関数演算から 2^{36} 回の SHA-0 の圧縮関数演算に改良できることを示す.

まず, Wang らの手法をステップ 20 までの衝突条件に適用して, ステップ 21 から 25 の衝突条件に対しては, 我々が提案する SM を適用する. ここで, ステップ 21 からステップ 25 までには以下の 4 つの条件があり, 以降これらの条件を満たすための SM を説明する: $a_{21,4} = a_{20,4}$ (or $a_{21,4} \neq a_{20,4}$), $a_{22,2} = m_{21,2}, a_{22,4} = a_{21,4}$ (or $a_{22,4} \neq a_{21,4}$), $a_{23,2} = m_{22,2}$.

Modification 1 まず, 以下の Extra 条件を満たすようにセットする: $a_{6,6} = m_{5,6}, m_{6,11} \neq m_{5,6}, m_{7,6} = m_{5,6}, a_{7,4} = 0, a_{8,4} = 1, m_{10,4} \neq m_{5,6}$. そして, 以下ようにメッセージを修正した場合, 衝突条件 $a_{21,4} = a_{20,4}$ (または, $a_{21,4} \neq a_{20,4}$) を確率ほぼ 1 で満たすことができる.

$$m_5 \leftarrow m_5 \oplus 2^5, m_6 \leftarrow m_6 \oplus 2^{10}, m_7 \leftarrow m_7 \oplus 2^5,$$
$$m_{10} \leftarrow m_{10} \oplus 2^3$$

そして, Extra 条件に対しても, パタン 2 を用いて衝突条件を満たす方法と同様の方法で前もって Extra 条件を満たすようにメッセージを生成しておく.

Remark 1 我々は, この MM を用いるとターゲットとなる衝突条件を他の衝突条件を壊すことなく確率ほぼ 1 で満たすことを計算機実験により確認した. また, この MM による計算量は 2 ステップ分以下の演算量となる.

Modification 2 ここでは, Extra 条件として以下の条件をセットする: $a_{11,21} = m_{10,21}, m_{11,26} \neq m_{10,21}, a_{10,23} = a_{9,23}, a_{12,19} = 0, a_{13,19} = 1, m_{15,19} \neq m_{10,21}, m_{19,26} \neq m_{18,21}$. そして, この条件を満たすメッセージを以下の方法により生成すると, 衝突条件 $a_{22,2} = m_{21,2}$ を確率ほぼ 1 で満たすことができる.

$$m_{10} \leftarrow m_{10} \oplus 2^{20}, m_{11} \leftarrow m_{11} \oplus 2^{25}, m_{15} \leftarrow m_{15} \oplus 2^{18}$$

ここで, この修正はパタン 1 を用いる.

Remark 2 δx を MM を行った時にある値 x に発生する差分値とすると, 差分を入力後, メッセージ拡大によって, 以下のメッセージ差分が出現する: $\delta m_{18} = \pm 2^{18} \pm 2^{20}, \delta m_{19} = \pm 2^{25}$ and $\delta m_{21} = \pm 2^{18} \pm 2^{20}$. ここで, Extra 条件として $m_{19,26} \neq m_{18,21}$ をセットすると, m_{18} の差分 $\pm 2^{20}$ の符号と m_{19} の $\pm 2^{25}$ の符号は逆になる. よって, δa_{19} と δm_{18} の符号の向きは同じになるため, $a_{19} \lll 5$ の差分 $\pm 2^{25}$ は差分 $\delta m_{19} = \pm 2^{25}$ によって打ち消される. 以上より, 新たに Extra 条件を設けることで, 他の条件に差分の影響が及ぼされる確率を小さくすることができる.

Remark 3 我々はこの MM が他の衝突条件に影響を与えずにターゲットの衝突条件を確率 97.5% で満たすことができることを計算機実験により確認した. また, この MM の計算量は 3 回のステップ関数演算以下で行うことができる.

Modification 3 まず, 以下の Extra 条件が成り立つと仮定する: $a_{11,8} = m_{10,8}, m_{11,13} \neq m_{10,8}, a_{10,10} = a_{9,10}, a_{12,6} = 0, a_{13,6} = 1, m_{15,6} \neq m_{10,8}, m_{19,13} \neq m_{18,8}$. そして, パタン 2

の message 修正法を用いた以下の修正を用いることで, $a_{22,4} = a_{21,4}$ (または, $a_{22,4} \neq a_{21,4}$) の衝突条件を確率ほぼ 1 で満たすことができる.

$$m_{10} \leftarrow m_{10} \oplus 2^7, m_{11} \leftarrow m_{11} \oplus 2^{12}, m_{15} \leftarrow m_{15} \oplus 2^5$$

Remark 4 我々はこの MM が他の衝突条件に影響を与えずにターゲットの衝突条件を確率ほぼ 100% で満たすことができることを計算機実験により確認した. また, この MM の計算量は 3 回のステップ関数演算以下で行うことができる.

Modification 4 まず, 以下の Extra 条件が成り立つと仮定する: $a_{11,16} = m_{10,16}, m_{11,21} \neq m_{10,16}, m_{12,16} \neq m_{10,16}, a_{12,14} = 0, a_{13,14} = 1, m_{15,14} \neq m_{10,16}, m_{19,21} \neq m_{18,16}$. そして, パタン 1 の message 修正法を用いた以下の修正を用いることで, $a_{23,2} = m_{22,2}$ の衝突条件を確率ほぼ 1 で満たすことができる.

$$m_{10} \leftarrow m_{10} \oplus 2^{15}, m_{11} \leftarrow m_{11} \oplus 2^{20}, m_{12} \leftarrow m_{12} \oplus 2^{15},$$
$$m_{15} \leftarrow m_{15} \oplus 2^{13}$$

Remark 5 我々はこの MM が他の衝突条件に影響を与えずにターゲットの衝突条件を確率ほぼ 97% で満たすことができることを計算機実験により確認した. また, この MM の計算量は 4 回のステップ関数演算以下で行うことができる.

7.5.3.1 SHA-0 のコリジョン探索の計算量

本章では SM を用いた SHA-0 のコリジョン探索の計算量を求める.

第 1 ブロックと第 2 ブロックのステップ 1-13. これらのステップのメッセージ生成の計算量は無視できる計算量である. これらの計算量については文献 [62] を参照されたい.

第 2 ブロックのステップ 14-20. このステップの値についている全ての衝突条件を満たすメッセージを生成するための計算量は 8 ステップ以下である.

第 2 ブロックのステップ 21. ステップ 21 の全ての衝突条件を満たすメッセージを生成するための計算量は以下の通りである.

$$8 + 1 + \frac{1}{2} \cdot 2 = 10.$$

第 2 ブロックのステップ 22. ステップ 22 の衝突条件を満たすメッセージを生成するための計算量の見積もりは以下のとおりである. ここで, $X_{22,i}$ をステップ 22 より前のステッ

プの衝突条件に対して m_{14}, m_{15} を i 回ランダムに選んでことで満たすために必要な計算量とする. この場合, 以下の式が成り立つ. ここで, $X_{22,0} = 0$ である.

$$X_{22,i} = \left(\frac{1}{2} \cdot 0.025\right)^{i-1} \cdot \left(10 + 1 + \frac{1}{2} \cdot 3 + \frac{1}{2} \cdot 3\right) + X_{22,i-1}$$

そして, $\lim_{i \to \infty} X_{22,i} \approx 15$ なので, このステップの計算量はおよそ 15 ステップ分である.

第 2 ブロックのステップ 23. ステップ 23 の衝突条件を満たすメッセージを生成する計算量は以下の通りである. ここで, $X_{23,i}$ をステップ 23 より前のステップの衝突条件に対して m_{14}, m_{15} を i 回ランダムに選んでことで満たすために必要な計算量とする. この場合, 以下の式が成り立つ. ここで, $X_{23,0} = 0$ である.

$$X_{23,i} = \left(\frac{1}{2} \cdot 0.03\right)^{i-1} \cdot \left(15 + 1 + \frac{1}{2} \cdot 4\right) + X_{23,i-1}$$

ここで, $\lim_{i \to \infty} X_{23,i} \approx 18$ なので, この計算量は 18 ステップ分となる.

第 2 ブロックのステップ $i(i = 24 - 80)$. Y_{i-1} をステップ $i - 1$ までの全ての衝突条件を満たすために必要な計算量とする. ここで, i ステップ目に n_i 個の衝突条件がある場合, 全ての条件を満たす確率は 2^{-n_i} である. よって, $Y_i = (Y_{i-1} + 1) \cdot 2^{n_i}$ となことから, $Y_{80} = 6180766429108$ となり, これは 2^{36} SHA-0 の圧縮関数演算である.

第 8 章

結論

本研究では，ハッシュ関数の安全性である IFRO 安全性に注目し，ハッシュ関数の実装性能の向上, 既存ハッシュ関数の救済, 既存ハッシュ関数の攻撃の 3 点について貢献した．具体的には，以下の 5 つのことを示した．

1. 重要なハッシュ関数である MD ハッシュ関数と重要な暗号アルゴリズム $\mathcal{C}$ である FDH 署名, PSS 署名，OAEP 暗号，RSA-KEM と, これらの安全性 $\mathcal{G}_s$ である EUF-CMA と IND-CCA に対し, $\mathcal{C}^{\mathsf{MD}} \succ_{\mathcal{G}_s} \mathcal{C}^{\mathcal{RO}}$ を示した．
2. H^{P} を重要なハッシュ関数である Sponge 関数と ChopMD とし, MS で定義される重要な安全性 $\mathcal{G}_m$ である CDA と実用的な暗号アルゴリズム $\mathcal{C}$ である EwH と REwH に対して，$\mathcal{C}^{\mathsf{H}^{\mathsf{P}}} \succ_{\mathcal{G}_m} \mathcal{C}^{\mathcal{RO}}$ を示した．
3. Sponge が IFRO 安全性を担保したまま高速化できることを示した．
4. 省リソース環境で実装可能な倍ブロック長ハッシュ関数において，世界で初めて IFRO 安全となる方式を提案した．
5. ハッシュ関数の強力な攻撃法である差分攻撃をし, 差分攻撃の限界点の検討を行った．

MD5 や SHA-1 が破られて以降，MD5 や SHA-1 と同じ思想で設計されている米国標準ハッシュ関数である SHA-256 や SHA-512 の安全性を疑問視する研究者が多くいた．本論文で示した 1 つ目の結果は，SHA-256 や SHA-512 が十分使えることを示し，この疑念を取り除く重要な結果であると考えている．

2 つ目の成果では，近年注目されている複数の攻撃者が結託することができる強力な攻撃を想定した MS の安全性で米国標準ハッシュ関数が安全に使えることを示した．この結果は，今後，MS の安全性を想定した暗号アルゴリズムの普及を促進する 1 つのきっかけとなると考えている．

Sponge は次世代米国標準ハッシュ関数である SHA-3 や他の実用的なハッシュ関数のモードである. すなわち, 3 つ目の結果は，SHA-3 に高速化できること示しており, SHA-3 の仕様に影響を与える可能性のある重要な成果であると考えている.

近年，IC チップや RFID タグ上など省リソース環境で暗号を実装するケースが増えており，4 つ目の結果は省リソース環境に適した方式であり，今後，多くの省リソース環境で使われる可能性がある.

差分攻撃法はハッシュ関数の強力な攻撃法であり, この限界点を見極めることが重要である. 5 つめの成果は, 差分攻撃法の限界点を見極める第 1 歩となる成果であると考えている.

最後に，本論文では，暗号を使うユーザに安心安全を与えるとともに，ユーザが暗号を快適に使える方式を提案することができ，暗号を使う全てのユーザにメリットのある成果を与えることができたと考えている.

付録

誕生日解析

補題 27 $\mathcal{F} : \{0,1\}^* \to \{0,1\}^n$ をランダム関数とする. $\mathcal{F}$ の呼び出し回数を q 回とすると, $\mathcal{F}(m) = \mathcal{F}(m')$ となる異なる 2 つの入力値 m, m' を見つけることができる確率は $q(q-1)/2^{n+1}$ 以下となる.

証明. $i \in \{1, \ldots, q\}$ を固定する. i 番目の $\mathcal{F}$ の入力値に対する出力値を z_i と書くことにする. z_i が $z_1, \ldots, z_{i-1}$ のいずれかの値と同じになる確率は $(i-1)/2^n$ 以下となる. そして, i を 1 から q まで足しあげると, $\mathcal{F}(m) = \mathcal{F}(m')$ となる異なる 2 つの入力値 m, m' を見つけることができる確率は下の確率以下となる.

$$\sum_{i=1}^{q} \frac{i-1}{2^n} = \frac{q(q-1)}{2^{n+1}}$$

■

PRF/PRP Switching Lemma [10]

補題 28 $\mathcal{F} : \{0,1\}^n \to \{0,1\}^n$ をランダム関数, $\mathcal{P} : \{0,1\}^n \to \{0,1\}^n$ をランダム置換とする. q 回オラクルにクエリして 1 bit の値を出力する任意の識別者 $\mathcal{D}$ に対して, 以下の式が成り立つ.

$$\Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1] \leq \frac{q(q-1)}{2^{n+1}}$$

証明. $\mathcal{F}$ の出力で衝突が起こるイベントを Coll とすると, 以下の式が成り立つ.

$$\begin{aligned}\Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1] &= \big(\Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1 \wedge \neg\mathsf{Coll}] + \Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1 \wedge \mathsf{Coll}]\big) - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1] \\ &\leq \Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1 | \neg\mathsf{Coll}] + \Pr[\mathsf{Coll}] - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1]\end{aligned}$$

ここで, $\mathsf{Coll} = \mathsf{false}$ の場合, $\mathcal{F}$ はランダム置換として振る舞うため, 以下の式が成り立つ.

$$\Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1 | \neg\mathsf{Coll}] - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1] = 0$$

よって, 以下の式が成り立つ.

$$\Pr[\mathcal{D}^{\mathcal{F}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathcal{P}} \Rightarrow 1] \leq \Pr[\mathsf{Coll}]$$

次に, $\Pr[\mathsf{Coll}]$ を評価する. ここで, $i \in \{1, \ldots, q\}$ を固定する. $\mathcal{D}$ が $\mathcal{F}$ と対話している場合, $\mathcal{D}$ の i 番目のクエリで $\mathcal{F}$ の出力に衝突が起こる確率は $(i-1)/2^n$ 以下となる. そして, i を 1 から q まで足しあげると, 以下の式が成り立つ.

$$\Pr[\mathsf{Coll}] \leq \sum_{i=1}^{q} \frac{i-1}{2^n} = \frac{q(q-1)}{2^{n+1}}$$

■

SMD 構造の衝突困難性の証明

定理 8.1 h が衝突困難性を満たすならば SMD^h も衝突困難性を満たす.

証明. 定理の対偶を取り, SMD^h の衝突困難性を破る攻撃者 $\mathcal{B}$ の存在を仮定して, h の衝突困難性を破る攻撃者 $\mathcal{A}$ の存在を証明する.

$\mathcal{B}$ が $\mathsf{SMD}^h(m) = \mathsf{SMD}^h(m')$ となる 2 つの異なる値 m, m' を出力したとする. この場合, m, m' について以下の場合分けを行う.

1. $|m| = |m'|$
2. $|m| \neq |m'|$

以下, それぞれのケースで $h(x) = h(x')$ となる異なる 2 つの値 x, x' を見つける h の衝突困難性を破る攻撃者 $\mathcal{A}$ が構成できることを証明する.

まず, $|m| = |m'|$ の場合を考える. この場合, $\mathsf{SMD}^h(m)$ と $\mathsf{SMD}^h(m')$ の計算で呼び出される h の呼び出し回数は等しくなる. ここで, $\mathsf{SMD}^h(m)$ と $\mathsf{SMD}^h(m')$ の中で最後に呼ばれる h から順番に IV に向かって h の入出力値をチェックしていくと, m, m' は $\mathsf{SMD}^h(m) = \mathsf{SMD}^h(m')$ となる 2 つの異なる値なので, 必ず, $x \neq x'$ かつ $h(x) = h(x')$ となる x, x' が存在する. この x, x' を $\mathcal{A}$ の出力値とすると, $\mathcal{A}$ は h の衝突困難性を破る攻撃者となる.

次に, $|m| \neq |m'|$ の場合を考える. ここで, $|\mathsf{pad}(m)| < |\mathsf{pad}(m')|$ とする. $|\mathsf{pad}(m)| > |\mathsf{pad}(m')|$ の場合は, m と m' を入れ替えて考えると同様の議論で証明できる. ここで, $\mathsf{pad}(m')$ の後ろの $|\mathsf{pad}(m)|$ bit を m_2, 残りの前半の値を m_1 と書くことにする. すなわち, $\mathsf{pad}(m) = m_1 \| m_2$ となる. pad は suffix-free 関数なので, $\mathsf{pad}(m) \neq m_2$ となる. そして, $\mathsf{SMD}^h(m)$ と $\mathsf{SMD}^h(m')$ の中で最後に呼ばれる h から順番に IV に向かって h の入出力値をチェックしていくと, $\mathsf{pad}(m') \neq m_2$ なので, 必ず $x \neq x'$ かつ $h(x) = h(x')$ となる x, x' が存在する. この x, x' を $\mathcal{A}$ の出力値として出力すると, $\mathcal{A}$ は h の衝突困難性を破る攻撃者となる.

■

IFRO 安全な定義域拡張構造

IFRO 安全な定義域拡張構造の設計方法を説明して, IFRO 安全性証明の例として, 簡略化した ChopMD 構造を用いたハッシュ関数の IFRO 安全性を証明する.

IFRO 安全な定義域拡張構造の設計方法

IFRO 安全な H^{P} を設計したい場合, (L, R) と対話する任意の識別者 $\mathcal{D}$ が Real World である $(L, R) = (\mathsf{H}^{\mathsf{P}}, \mathsf{P})$ と Ideal World である $(L, R) = (\mathcal{RO}, S)$ が識別できないようなシミュレータ S が構成できるかどうかを考える必要がある.

そして, S を構成する場合, 次の 2 つの S の条件を満たすか否かを確認する必要がある.

- S の条件 1: まず, R に関するシミュレートで, Real World では $R = \mathsf{P}$ で, Ideal World では $R = S$ なので, S が P 自信をシミュレートできることを確認する必要がある. 例えば, P がランダムオラクル圧縮関数 h の場合, 初出のクエリに対する出力はランダムに選ばれ, 繰り返しのクエリに対する出力は過去に出力した値と同じ値となるため, S はこの性質をシミュレートする必要がある.
- S の条件 2: L の入出力値と R の入出力値の関係のシミュレートで, Real World で

は, $(L, R) = (\mathsf{H}^{\mathsf{P}}, \mathsf{P})$ で, H^{P} は出力値を H の構造に従って P を用いて計算しているので, (L, R) の入出力に H の構造に依存した関係がある. 一方で, Ideal World では $(L, R) = (\mathcal{RO}, S)$ で $\mathcal{RO}$ は H の構造とは関係なく単独で動作する関数なので, S は $(\mathcal{RO}, S)$ の入出力に H の構造に依存した関係があるようにシミュレートする必要がある.

$\mathcal{D}$ は (L, R) と対話して, P の仕様と H 仕様をもとに, (L, R) の入出力値から Real World と Ideal World を識別しようとするが, 条件 1 は P の仕様に関するシミュレートで, 条件 2 は H の仕様に関するシミュレートで, この 2 つの条件を満足する S が構成できれば H^{P} が IFRO 安全となることが証明できる.

IFRO 安全性証明の例

IFRO 安全性証明を簡易版の ChopMD 構造を用いたハッシュ関数を例に説明する. ここで考えるハッシュ関数は, P としてランダムオラクル圧縮関数 $h : \{0,1\}^{r+n} \to \{0,1\}^n$ を用いて, H として入力長が r bit と $2r$ bit に制限して, pad 関数を取り除いて, $chop_s$ 関数として $s = n/2$ とする簡易版 ChopMD 構造を用いたハッシュ関数 ChopMD^h を考える. ChopMD^h の入力長は r bit または $2r$ bit で, 出力長は $n/2$ bit である.

まず, 以下にシミュレータ S を定義する. S は h をシミュレートするため, S の入出力長は h と同じになるように定義する. そして, 2 つの S の条件を満たすように S の手続きを定義する. 条件 1 に対応する S の振る舞いとして, 初出のクエリに対して出力をランダムに選んで, テーブル $\mathcal{T}$ を導入して, このクエリと出力値を $\mathcal{T}$ に記録しておく. そして, 繰り返しのクエリが来た場合, $\mathcal{T}$ を参照して, 過去に出力した値と同じ値を出力する. 次に, 2 つ目の条件について, Real World では, ChopMD の構造から, r bit の m_1 に対して, $z_1 = L(m_1)$, $y_1 = R(IV, m_1)$ ならば $z_1 = y_1[s+1, n]$ となり, r bit の m_2 に対して, $z_2 = L(m_1 \| m_2)$, $y_2 = R(y_1, m_2)$ ならば $z_2 = y_2[s+1, n]$ となるため, Ideal World でも同じ関係が (L, R) の入出力で成り立つように S を定義する. この 2 つの条件に注意して, x を n bit の値, m を r bit として, S へのクエリ (x, m) に対して, 以下の手続きにより n bit の出力値 y を定義する.

1. If $\exists(x, m, y) \in \mathcal{T}$ then return y
2. Else
   (a) If $x = IV$ then $y[1, s] \xleftarrow{\$} \{0,1\}^s$; $y[s+1, n] \leftarrow \mathcal{RO}(m)$
   (b) Else If $\exists(IV, m', x) \in \mathcal{T}$ s.t. there exists just one such triple then $y[1, s] \xleftarrow{\$} \{0,1\}^s$; $y[s+1, n] \leftarrow \mathcal{RO}(m' \| m)$

(c) Else $y \xleftarrow{\$} \{0,1\}^n$
(d) $\mathcal{T} \xleftarrow{\cup} (x, m, y)$
(e) Return y

ここで, ステップ 1 より, 繰り返しクエリに対して $\mathcal{T}$ を参照して過去に返した値と同じ値を出力して, ステップ 2 以降の手順で出力値をランダムに選んでいるため, 条件 1 を満足する. そして, ステップ a より, r bit の m_1 に対して, $z_1 = L(m_1)$, $y_1 = R(IV, m_1)$ ならば $z_1 = y_1[s+1, n]$ となり, ステップ b より, r bit の m_2 に対して, $z_2 = L(m_1 \| m_2)$, $y_2 = R(y_1, m_2)$ ならば $z_2 = y_2[s+1, n]$ となるため, 条件 2 を満足する.

次に, この S に対して任意の $\mathcal{D}$ が Real World である $(L, R) = (\mathsf{H}^{\mathsf{P}}, \mathsf{P})$ と Ideal World である $(L, R) = (\mathcal{RO}, S)$ を識別できないことを証明する. 具体的には, $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{ChopMD}^h, \mathcal{RO}, S}(\mathcal{D})$ が無視できる値で抑えられることを, 以下の 3 つのゲームを用いて証明する. 以下, それぞれのゲームで識別者が (L, R) と対話する.

- Game 1: このゲームは Real World である. すなわち, $(L, R) = (\mathsf{ChopMD}^h, h)$ となる.
- Game 2: Game 1 からの変更点は, R が h から S に変わった点である. すなわち, $(L, R) = (\mathsf{ChopMD}^S, S)$ となる.
- Game 3: このゲームは Ideal World である. Game 2 からの変更点は, L が ChopMD^S から $\mathcal{RO}$ に変わった点である. すなわち, $(L, R) = (\mathcal{RO}, S)$ となる.

ここで, Game i で $\mathcal{D}$ が 1 を出力するイベントを G_i と書くことにすると, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{ChopMD}^h, \mathcal{RO}, S}(\mathcal{D}) = \Pr[G_1] - \Pr[G_3] = \Pr[G_1] - \Pr[G_2] + \Pr[G_2] - \Pr[G_3]$$

そして, $\Pr[G_1] - \Pr[G_2]$ と $\Pr[G_2] - \Pr[G_3]$ が無視できる値で抑えることができれば, 証明完了となる.

$\Pr[G_1] - \Pr[G_2]$ の評価: Game 1 と Game 2 では R オラクルが異なっており, Game 1 では $R = h$, Game 2 では $R = S$ である. ここで, S は前章で述べた条件 1 を満たすように定義されているため, Game 1 と Game 2 で $\mathcal{D}$ の振る舞いは変わらない. 具体的には, S は初出のクエリに対して出力はランダムに選び, 繰り返しのクエリに対しては過去に出力した値と同じ値を返すように定義されているため, Game 1 と Game 2 の違いは $\mathcal{D}$ の振る舞いに影響を与えない. よって, $\Pr[G_1] - \Pr[G_2] = 0$ となる.

$\Pr[G_2] - \Pr[G_3]$ **の評価:** Game 2 と Game 3 では L オラクルが異なっており, Game 2 では $L = \mathsf{ChopMD}^S$, Game 3 では $L = \mathcal{RO}$ となるため, Game 2 で L へのクエリ m に対して, $L(m)$ が $\mathcal{RO}(m)$ と等しいことを示す必要がある. そして, Game 2 では, L へのクエリ m に対して, $L(m)$ は ChopMD の構造に従って S を用いて定義されるため, L の入出力値と R の入出力値に ChopMD の構造に依存した関係がある. よって, Game 3 で, L の入出力値と R の入出力値が同様の関係を持つことを示す必要がある. そして, S は前章で述べた条件 2 を満たすように定義されているため, Game 2 と Game 3 で $\mathcal{D}$ の振る舞いは無視できる確率を除いて変わらないことが示せる. 具体的には, 以下の 2 点が示せれば Game 2 と Game 3 で識別者の振る舞いが同じとなることが言える.

- Point 1: Game 2 で任意の L へのクエリ m に対して, 出力値が $\mathcal{RO}(m)$ と等しくなること.
- Point 2: Game 3 で任意の r bit の値 m_1, m_2 に対して, L へのクエリ m_1 に対する出力値 z_1 と R へのクエリ (IV, m_1) に対する出力値 y_1 に対して, $y_1[s+1, n] = z_1$ となり, L へのクエリ $m_1 \| m_2$ に対する出力値 z_2 と R へのクエリ (y_1, m_2) に対する出力値 y_2 に対して, $y_2[s+1, n] = z_2$ となること.

ここで, 上記の 2 点が成り立つことを示すために, 以下の補助定理を用いる.

補題 29 以下のイベントが起こらなければ,
任意の S へのクエリレスポンス (IV, m, y) に対して $y[s+1, n] = \mathcal{RO}(m)$ となり,
任意の S へのクエリレスポンス $(IV, m_1, y_1), (y_1, m_2, y_2)$ に対して $y_2[s+1, n] = \mathcal{RO}(m_1 \| m_2)$ となる.
$y = S(IV, m), y_1 = S(IV, m_1), y_2 = S(y_1, m_2)$ であることに注意されたい.
ここで, $\mathcal{T}_i^*$ を S が i 回呼び出される前の全ての S のクエリレスポン (x, m, y) の $x[1, s]$, $y[1, s]$ を記録したテーブルとして, イベント Bad を定義する.

- Bad: ある i に対して, S への i 番目のクエリ (x_i, m_i) に対して, その出力値 y_i が $y_i[1, s] \in \mathcal{T}_i^* \cup \{x_i[1, s]\}$ となるイベント.

証明. Bad が起こらないと仮定する.

まず, S の構造から任意の S へのクエリレスポンス (IV, m, y) に対して $y[s+1, n] = \mathcal{RO}(m)$ となる.

次に, 任意の S へのクエリレスポンス $(IV, m_1, y_1), (y_1, m_2, y_2)$ に対して, Bad は起こらないと仮定しているので, (IV, m_1, y_1) が定義された後で (y_1, m_2, y_2) が定義される. そして,

Bad は起こらないと仮定しているので, S の出力でコリジョンは起こらないため, S へのクエリ (y_1, m_2) に対して b が実行されて $y_2[s+1, n] = \mathcal{RO}(m_1 \| m_2)$ となる.

■

上記の補助定理より, Bad が Game 2 で起こらなければ $L = \mathsf{ChopMD}^S$ なので Point 1 が成り立ち, そして, Bad が Game 3 で起こらなければ Point 2 が成り立つ. よって, Game 2 で Bad が起こるイベントを Bad_2, Game 3 で Bad が起こるイベントを Bad_3 と書くことにすると, 以下の式が成り立つ.

$$\Pr[G_2] - \Pr[G_3] \leq \Pr[Bad_2] + \Pr[Bad_3]$$

まず, $\Pr[Bad_2]$ を評価する. ここで, Game 2 で, S が呼び出される回数を q_2 とする. ここで, i を固定して, S への i 番目のクエリ (x_i, m_i) の出力値 y_i の $y_i[1, s]$ は S によって $y_i[1, s] \in \mathcal{T}_i^* \cup \{x_i[1, s]\}$ とは独立にランダムに選ばれるため, $y_i[1, s] \in \mathcal{T}_i^* \cup \{x_i[1, s]\}$ となる確率は $(2(i-1)+1)/2^s$ となる. よって, 以下の式が成り立つ.

$$\Pr[G_2] \leq \sum_{i=1}^{q_2} \frac{2(i-1)+1}{2^s} = \frac{q_2^2}{2^s}$$

次に, $\Pr[Bad_3]$ を評価する. ここで, Game 3 で, S が呼び出される回数を q_3 とする. この評価は, $\Pr[Bad_2]$ の解析と同様の解析方法が適用できて, 以下の式が成り立つ.

$$\Pr[G_3] \leq \sum_{i=1}^{q_3} \frac{2(i-1)+1}{2^s} = \frac{q_3^2}{2^s}$$

以上より, 以下の式が成り立つ.

$$\Pr[G_2] - \Pr[G_3] \leq \frac{q_3^2 + q_2^2}{2^s}$$

以上の議論から, 以下の式が成り立つ.

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{ChopMD}^h, \mathcal{RO}, S}(\mathcal{D}) \leq \frac{q_3^2 + q_2^2}{2^s}$$

ここで, $q_2 \ll 2^{s/2}, q_3 \ll 2^{s/2}$ の場合, $\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{ChopMD}^h, \mathcal{RO}, S}(\mathcal{D})$ は無視できるくらい小さい値となり, ChopMD^h が IFRO 安全性を満たす.

参考文献

[1] Masayuki Abe, Eike Kiltz, and Tatsuaki Okamoto. Chosen-Ciphertext Security with Optimal Ciphertext Overhead. In *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 355–371. Springer, 2008.

[2] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. In *CHES*, volume 6225 of *LNCS*, pages 1–15. Springer, 2010.

[3] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and Efficiently Searchable Encryption. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552. Springer, 2007.

[4] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In *ASIACRYPT*, volume 5912 of *LNCS*, pages 232–249. Springer, 2009.

[5] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles. In *CRYPTO*, volume 5157 of *LNCS*, pages 360–378. Springer, 2008.

[6] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-Locked Encryption and Secure Deduplication. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

[7] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[8] Mihir Bellare and Phillip Rogaway. Optimal Asymmetric Encryption – How to Encrypt with RSA. In *EUROCRYPT*, volume 950 of *Lecture Notes in Computer Science*,

pages 92–111. Springer, 1994.

[9] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[10] Mihir Bellare and Phillip Rogaway. Code-Based Game-Playing Proofs and the Security of Triple Encryption. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *Ecrypt Hash Workshop*, 2007.

[12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In *EUROCRYPT*, pages 181–197, 2008.

[13] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak SHA-3 submission. Submission to NIST (Round 3). 2011.

[14] Eli Biham and Rafi Chen. Near-collisions of SHA-0. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 290–305, 2004.

[15] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and reduced SHA-1. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 36–57, 2005.

[16] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Functions Constructions from PGV. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 103–118. Springer, 2002.

[17] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 312–325, 2011.

[18] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On Notions of Security for Deterministic Encryption, and Efficient Constructions without Random Oracles. In *CRYPTO*, volume 5157 of *LNCS*, pages 335–359. Springer, 2008.

[19] Dan Boneh. Simplified OAEP for the RSA and Rabin functions. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2001.

[20] Bruno O. Brachtl, Don Coppersmith, Myrna M. Hyden, Stephen M. Matyas Jr, Carl H. W. Meyer, Jonathan Oseas, Shaiy Pilpel, and Michael Schilling. Data authentication using modification detection codes based on a public one way encryption function. US Patent No. 4,908,861, 1990 (filed August 28, 1987).

[21] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *CoRR*, cs.CR/0010019, 2000.

[22] Florent Chabaud and Antoine Joux. Differential collisions in SHA-0. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 56–71, 1998.

[23] Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. Indifferentiable Security Analysis of Popular Hash Functions with Prefix-Free Padding. In *ASIACRYPT*, volume 4284 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2006.

[24] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

[25] Ivan Damgård. A Design Principle for Hash Functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[26] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In *EUROCRYPT (Full Version in ePrint 2009/177)*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009.

[27] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In *ePrint 2009/177*, 2009.

[28] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009.

[29] Ewan Fleischmann, Michael Gorski, and Stefan Lucksl. Security of Cyclic Double Block Length Hash Functions. In *IMA Int. Conf*, volume 5921 of *Lecture Notes in Computer Science*, pages 153–175. Springer, 2009.

[30] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. In *CRYPTO*, Lecture Notes in Computer Science, pages 260–274. Springer, 2001.

[31] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A Unified Approach to Determin-

istic Encryption: New Constructions and a Connection to Computational Entropy. In *TCC2012*, pages 582–599, 2012.

[32] Zheng Gong, Xuejia Lai, and Kefei Chen. A synthetic indifferentiability analysis of some block-cipher-based hash functions. In *Des. Codes Cryptography 48*, pages 293–305, 2008.

[33] Jian Guo, Thomas Peyrin, and Axel Poschmann. The photon family of lightweight hash functions. In *CRYPTO*, volume 6841 of *LNCS*, pages 222–239. Springer, 2011.

[34] Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[35] Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In *ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007.

[36] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, pages 315–324, 2007.

[37] Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In *EUROCRYPT*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.

[38] Jooyoung Lee and Martijn Stam. Mjh: A faster alternative to mdc-2. In *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236. Springer, 2011.

[39] Stefan Lucks. A collision-resistant rate-1 double-block-length hash function. In *Symmetric Cryptography*, Symmetric Cryptography, Dagstuhl Seminar Proceedings 07021, 2007.

[40] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[41] Ralph C. Merkle. One Way Hash Functions and DES. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[42] Carl H. W. Meyer and Michael Schilling. Chargement securise d'un programma avec code de detection. 1987.

[43] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental Deterministic Public-Key Encryption, (Full Version in ePrint 2012/047). In *EUROCRYPT*, 2012.

[44] Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, and Kazuo Ohta. Improved collision search for SHA-0. In *Advances in Cryptology - ASIACRYPT 2006, 12th International Conference on the Theory and Application of Cryptology and Information Security, Shanghai, China, December 3-7, 2006, Proceedings*, pages 21–36, 2006.

[45] National Institute of Standards and Technology. DRAFT FIPS PUB 202. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions .

[46] National Institute of Standards and Technoloty. FIPS PUB 180-4 Secure Hash Standard. In *FIPS PUB*, 2012.

[47] Onur Özen and Martijn Stam. Another Glance at Double-Length Hashing. In *IMA Int. Conf*, volume 5921 of *Lecture Notes in Computer Science*, pages 176–201. Springer, 2009.

[48] Duong Hieu Phan and David Pointcheval. Chosen-Ciphertext Security without Redundancy. In *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[49] Duong Hieu Phan and David Pointcheval. OAEP 3-Round: A Generic and Secure Asymmetric Encryption Padding. In *ASIACRYPT*, pages 63–77, 2004.

[50] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[51] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. In *EUROCRYPT (Full Version: ePrint 2011/339)*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

[52] Victor Shoup. A Proposal for an ISO Standard for Public Key Encryption (version 2.1). 2001.

[53] Victor Shoup. OAEP Reconsidered. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259. Springer, 2001.

[54] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[55] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. In *Cryptology ePrint Archive: 2004/332*, 2004. `http://eprint.iacr.org/2004/332`.

[56] Martijn Stam. Blockcipher-Based Hashing Revisited. In *FSE*, volume 5665 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2009.

[57] Gene Tsudik. Message authentication with one-way hash functions. In *INFOCOM*, pages 2055–2059, 1992.

[58] X. Wang, D. Feng, H. Chen, X. Lai, and X. Yu. Collision for hash functions md4, md5, haval-128 and ripemd. In *In Rump Session of CRYPTO 2004 and Cryptology ePrint Archive*, 2004.

[59] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 1–18, 2005.

[60] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 17–36, 2005.

[61] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 19–35, 2005.

[62] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient collision search attacks on SHA-0. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 1–16, 2005.

[63] Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky Random Oracle (Extended Abstract). In *ProvSec*, volume 5324 of *Lecture Notes in Computer Science*, pages 226–240. Springer, 2008.

謝辞

本論文は筆者が電気通信大学大学大学院情報理工学研究科総合情報学専攻博士後期課程に在籍中の研究成果をまとめたものである．同専攻教授 太田和夫 先生には指導教員として，本研究を行うにあたり，多岐にわたる助言と，丁寧なご指導を戴いた．ここに深謝の意を表する．

同専攻准教授 崎山一男 先生には副査としてご助言を戴くとともに本論文の細部にわたりご指導を戴いた．ここに深謝の意を表する．

東京大学大学院新領域創成科学研究科複雑理工学専攻准教授 國廣昇 先生は，筆者が学部・修士課程に在籍中，太田先生とともに丁寧にご指導を頂いた．ここに深謝の意を表する．

また, NTT セキュアプラットフォーム研究所の米山一樹 氏, 並びに, 佐々木悠 氏, Nanyang Technological University の Lei Wang 氏, 株式会社富士通研究所の下山武司 氏，並びに，矢嶋純 氏には，多くの議論の機会を与えて戴くとともに，有益なご助言を戴いた．ここに同氏らに対して感謝の意を表する．

三菱電機株式会社の粕谷智巳 氏には，博士後期課程に進学する機会を与えて戴くとともに，有益なご助言を頂いた．ここに深謝の意を表する．

最後に，筆者が博士後期課程に在学中に多岐にわたり協力し，支えてくれた 妻 優子 に深謝の意を表する．

本論文に関連した研究業績

査読付き論文誌

1. Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta. Security of Cryptosystems Using Merkle-Damgård in the Random Oracle Model. IEICE Transactions, 94-A(1):57–70, 2011.
2. Yusuke Naito, Kazuo Ohta, and Noboru Kunihiro. Improved Collision Search for Hash Functions: New Advanced Message Modification. IEICE Transactions, 91-A(1):46–54, 2008.

査読付き国際会議

1. Yusuke Naito and Kazuo Ohta. Improved Indifferentiable Security Analysis of PHOTON. In SCN, volume 8642 of Lecture Notes in Computer Science, pages 340–357. Springer, 2014.
2. Yusuke Naito, Kazuki Yoneyama, and Kazuo Ohta. Reset Indifferentiability from Weakened Random Oracle Salvages One-Pass Hash Functions. In ACNS, volume 8479 of Lecture Notes in Computer Science, pages 235–252. Springer, 2014.
3. Yusuke Naito. Blockcipher-Based Double-Length Hash Functions for Pseudorandom Oracles. In Selected Areas in Cryptography, volume 7118 of Lecture Notes in Computer Science, pages 338–355. Springer, 2011.
4. Yusuke Naito. Blockcipher-Based Double-Length Hash Functions as Random Oracles. In ECRYPT II Hash Workshop, 2011. This paper has been published at http://www.ecrypt. eu.org/hash2011/program.shtml.
5. Yusuke Naito, Kazuki Yoneyama, LeiWang, and Kazuo Ohta. How to Confirm Cryptosystems Security: The Original Merkle-Damgård Is Still Alive! In ASI-

ACRYPT, volume 5912 of Lecture Notes in Computer Science, pages 382–398. Springer, 2009.

6. Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Search for SHA-0. In ASIACRYPT, volume 4284 of Lecture Notes in Computer Science, pages 21–36. Springer, 2006.

全ての研究業績

査読付き論文誌

1. Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta. Security of Cryptosystems Using Merkle-Damgård in the Random Oracle Model. IEICE Transactions, 94-A(1):57–70, 2011.
2. Jun Yajima, Terutoshi Iwasaki, Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Thomas Peyrin, Noboru Kunihiro, and Kazuo Ohta. A Strict Evaluation on the Number of Conditions for SHA-1 Collision Search. IEICE Transactions, 92-A(1):87–95, 2009.
3. Yusuke Naito, Kazuo Ohta, and Noboru Kunihiro. Improved Collision Search for Hash Functions: New Advanced Message Modification. IEICE Transactions, 91-A(1):46–54, 2008.
4. Yu Sasaki, Yusuke Naito, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Attacks on MD4 and MD5. IEICE Transactions, 90-A(1):36–47, 2007.

査読付き国際会議

1. Yusuke Naito and Kazuo Ohta. Improved Indifferentiable Security Analysis of PHOTON. In SCN, volume 8642 of Lecture Notes in Computer Science, pages 340–357. Springer, 2014.
2. Yusuke Naito, Kazuki Yoneyama, and Kazuo Ohta. Reset Indifferentiability from Weakened Random Oracle Salvages One-Pass Hash Functions. In ACNS, volume 8479 of Lecture Notes in Computer Science, pages 235–252. Springer, 2014.
3. Yusuke Naito, Yu Sasaki, Lei Wang, and Kan Yasuda. Generic State-Recovery and

Forgery Attacks on ChopMD-MAC and on NMAC/HMAC. In IWSEC, volume 8231 of Lecture Notes in Computer Science, pages 83–98. Springer, 2013.

4. Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta. Security of Practical Cryptosystems Using Merkle-Damgård Hash Function in the Ideal Cipher Model. In ProvSec, volume 6980 of Lecture Notes in Computer Science, pages 281–296. Springer, 2011.
5. Yusuke Naito. Blockcipher-Based Double-Length Hash Functions for Pseudorandom Oracles. In Selected Areas in Cryptography, volume 7118 of Lecture Notes in Computer Science, pages 338–355. Springer, 2011.
6. Yusuke Naito. Blockcipher-Based Double-Length Hash Functions as Random Oracles. In ECRYPT II Hash Workshop, 2011. This paper has been published at http://www.ecrypt. eu.org/hash2011/program.shtml.
7. Yusuke Naito, Kazuki Yoneyama, LeiWang, and Kazuo Ohta. How to Confirm Cryptosystems Security: The Original Merkle-Damgård Is Still Alive! In ASIACRYPT, volume 5912 of Lecture Notes in Computer Science, pages 382–398. Springer, 2009.
8. Jun Yajima, Terutoshi Iwasaki, Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. A strict evaluation method on the number of conditions for the SHA-1 collision search. In ASIACCS, pages 10–20. ACM, 2008.
9. Jun Yajima, Yu Sasaki, Yusuke Naito, Terutoshi Iwasaki, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. A New Strategy for Finding a Differential Path of SHA-1. In ACISP, volume 4586 of Lecture Notes in Computer Science, pages 45–58. Springer, 2007.
10. Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Jun Yajima, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Search for SHA-0. In ASIACRYPT, volume 4284 of Lecture Notes in Computer Science, pages 21–36. Springer, 2006.
11. Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. How to Construct Sufficient Conditions for Hash Functions. In VIETCRYPT, volume 4341 of Lecture Notes in Computer Science, pages 243–259. Springer, 2006.
12. Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Attack on MD4 with Probability Almost 1. In ICISC, volume 3935 of Lecture Notes

in Computer Science, pages 129–145. Springer, 2005.

査読無し論文

1. 内藤祐介, 米山一樹, 太田和夫. マルチステージゲームでのランダムオラクルとの置き換えの再考. In SCIS, 2014. 2B4-1.
2. 内藤祐介. PHOTON の Indifferentiability について. In SCIS, 2013. 4B1-3.
3. 内藤祐介. 強識別不可能安全なハッシュ関数のマルチステージゲームでの置き換えについて. In SCIS, 2012. 4B2-5.
4. 内藤祐介. Blockcipher-based Double-length Hash Functions for Pseudorandom Oracles. In SCIS, 2011. 4B2-5.
5. 内藤祐介, 王磊, 米山一樹, 太田和夫. New Analysis of Davies-Meyer Merkle-Damgård. In SCIS, 2010. 4D1-2.
6. Yusuke Naito, Lei Wang and Kazuo Ohta. How to Construct Cryptosystems and Hash Functions in Weakened Random Oracle Models. In ISEC2009-26, pages 131–138, 2009.
7. 内藤祐介, 太田和夫, 王磊, 米山一樹. Merkle-Damgård 構造の強識別不可能性 (Indifferentiability) の再考. In SCIS, 2009. 2A4-6.
8. 内藤祐介, 太田和夫. eTCR 性と TCR 性の間の定義の考察. In CSS, 2008. D8-1.
9. 内藤祐介. eTCR 性ハッシュ関数の構成法の考察. In SCIS, 2008. 3A4-1.
10. 内藤祐介, 國廣昇, 太田和夫. ハッシュ関数のコリジョン探索の改良-新たな Advanced Message Modification の提案-. In SCIS, 2007. 1A1-1.
11. 岩崎輝星, 内藤祐介, 矢嶋純, 佐々木悠, 下山武司, 國廣昇, 太田和夫. Strategy for Selecting Disturbance Vector of SHA-1. In SCIS, 2007. 1A1-3.
12. 佐々木悠, 内藤祐介, 矢嶋純, 岩崎輝星, 下山武司, 國廣昇, 太田和夫. SHA-1 差分パス構築アルゴリズム. In SCIS, 2007. 1A1-4.
13. 矢嶋純, 佐々木悠, 岩崎輝星, 内藤祐介, 下山武司, 國廣昇, 太田和夫. SHA1 差分パス自動生成ツール. In SCIS, 2007. 1A1-5.
14. Yu Sasaki, Yusuke Naito, Jun Yajima, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta. How to Construct Sufficient Condition in Searching Collisions of MD5. In SCIS, 2006. 4E1-1.
15. 矢嶋純, 下山武司, 佐々木悠, 内藤祐介, 國廣昇, 太田和夫. MD5 のコリジョン探索にお

ける差分パスの構築法について- Wang の差分パスは最適か-. In SCIS, 2006. 4E1-2.
16. 内藤祐介, 佐々木悠, 下山武司, 矢嶋純, 國廣昇, 太田和夫. SHA-0 に対する Message Modification の考察. In SCIS, 2006. 4E1-3.
17. Yu Sasaki and Yusuke Naito and Noboru Kunihiro and Kazuo Ohta. Improved Collision Attack on MD5. In ISEC2005-67, pages 35–42, 2005.
18. 内藤祐介, 佐々木悠, 太田和夫, 國廣昇. MD4 に対するコリジョンアタックの改良. In ISEC2005- 58, pages 109–116, 2005.
19. 内藤祐介, 指田岳彦, 根岸大宙, 太田和夫, 國廣昇. TOYOCRYPT への故障利用攻撃. In SCIS, 2005. 2D2-4.

著者略歴

- 平成 17 年 3 月　電気通信大学 電気通信学部 情報工学科卒業
- 平成 17 年 4 月　電気通信大学 大学院電気通信学研究科 情報通信工学専攻博士前期課程入学
- 平成 19 年 3 月　同　修了
- 平成 19 年 4 月　三菱電機株式会社入社
- 平成 26 年 4 月　電気通信大学 大学院情報理工学研究科 総合情報学専攻博士後期課程入学 現在に至る