軽量暗号の安全性に関する調査及び評価
(Elephant, ISAP, Romulus)

井上明子 (NEC)

2022 年 12 月

目 次

第1章　エグゼクティブサマリー

現在開催中の NIST Lightweight cryptography プロジェクト (以下, NIST LWC) [1]においてファイナリストとして残っている 10 候補のうち, 3 つの候補 Elephant [21], ISAP [38], Romulus [55] の安全性について調査を行った. 詳細には, Final round[2]に提出されている各候補の仕様に対する安全性評価に関する文献を調査した. その結果として 2022 年 9 月末時点では, 全ての候補に対して提案者らが主張する安全性を損ねるような解析結果は発表されていないことが分かった.

本報告書では, 各候補に対してその仕様 (アルゴリズム) 及び安全性に関する文献の調査結果を記載する. まず 2 章で必要な記法等の準備を行い, 3 章, 4 章, 5 章でそれぞれ Elephant, ISAP, Romulus について述べる. 各候補の章は準備, 仕様, 安全性の 3 節に分かれている. 準備の節ではその候補の章に必要な記法を記載する. 仕様の節では, Final round に提出されている仕様書 [21,38,55] に基づき, 各候補のアルゴリズムを記載する. 但し, 1 つの候補に安全性や効率性等の異なる複数のアルゴリズムが含まれるため, そのうち Primary member として挙げられているアルゴリズムのみを詳述し, 他アルゴリズムは概要を記載する. 安全性の節は以下 4 つの構成になっている.

- **仕様上の安全性**: 各候補の提案者らが主張する主要な安全性とそのレベルを概説する.
- **モードの安全性**: 各候補が用いるモードの証明可能安全性について, 著者らの主張の詳細や安全性評価の調査結果をまとめる.
- **プリミティブの安全性及び関連方式の安全性**: 各候補で用いる暗号プリミティブの攻撃可能段数の調査結果をまとめる. 必要に応じてそのプリミティブを用いた他方式の攻撃可能段数についても述べる.
- **その他**: 上記 2 つ以外の安全性解析結果について述べる. 例として, プリミティブのラウンド数を削減した場合の各候補への安全性評価, サイドチャネル攻撃評価, 量子計算機を用いた場合の安全性評価が挙げられる.

注意事項として, ISAP においては NIST LWC ファイナリストの 1 つ Ascon [41] とアルゴリズムを共有する部分があるため, 該当部分のアルゴリズムや共通する安全性調査の結果は概要のみを記載することとする. 詳細は Ascon の安全性調査レポートを参照されたい.

[1] https://csrc.nist.gov/projects/lightweight-cryptography
[2] https://csrc.nist.gov/Projects/lightweight-cryptography/finalists

第2章　準備

2.1　記号

自然数 n に対し, $\{0,1\}^n$ は n ビットのビット列の集合を指し, $\{0,1\}^*$ は任意長のビット列の集合を表す. ビット列 $X \in \{0,1\}^*$ に対し, $|X|$ は X のビット長を指す. ビット列 $X, Y \in \{0,1\}^*$ に対し, $X \| Y$ は X と Y の連結であり, $X \oplus Y$ は X と Y の排他的論理和を表す. 集合 A に対し, $a \xleftarrow{\$} A$ は A から元 a を一様ランダムに選ぶ操作を指す.

2.2　認証暗号

入力データの秘匿と改ざん検知を同時に実現する共通鍵暗号方式を認証暗号 (Authenticated encryption with associated data; AEAD) と呼ぶ [12]. 秘密鍵を共有する二者間において, 認証暗号の暗号化関数の入力は秘密鍵 K, 初期ベクトルの役割を持つナンス (Nonce) N, 任意長の平文 M, 任意長の AD (Associated data) A である. ここで AD とは暗号化はしたくないが改ざん検知は行いたい情報であり, 例えば通信プロトコルのバージョンなどの情報がこれに該当する. 暗号化関数出力は暗号文 C とタグ T である. 一方, 復号関数の入力は秘密鍵 K, ナンス N, AD A, 暗号文 C, タグ T である. もし (N, A, C, T) に改ざんが検知されなければ, 復号関数は復号結果の平文 M を出力し, そうでなければ単一のエラーメッセージ $\perp$ を出力する.

安全性指標　認証暗号の安全性は秘匿 (Privacy) と改ざん検知 (Authenticity) の 2 つの指標によって評価される. 秘匿の安全性は, 暗号化オラクルにアクセスできる攻撃者が認証暗号の暗号化関数とランダムオラクル \$ を判別できる確率で評価される. 但し, \$ のインターフェースや入出力のビット長は暗号化関数と同一であるとする. 改ざん検知の安全性は, 暗号化オラクルと復号オラクルにアクセスできる攻撃者が復号オラクルから $\perp$ 以外の出力を得られる確率で評価される. この 2 つの安全性を定義する攻撃者の設定として「暗号化オラクルに対して同じナンスを 2 度以上クエリしてはならない」というものがあり, この設定は Nonce respecting と呼ばれる. この設定に従わない攻撃者, つまり暗号化クエリに対して同じナンスを 2 度以上クエリする攻撃者の設定を Nonce misuse と呼ぶ.

また, 秘匿と改ざん検知の 2 つの安全性を 1 つにまとめた安全性指標も存在する [95]. その安全性は, 暗号化オラクルと復号オラクルにアクセスできる攻撃者が Real world (認証暗号の暗号化関数及び復号関数) と Ideal world (\$ 及び如何なる入力に対しても $\perp$ のみを出力する関数) とを判別できる確率で評価される. この安全性は上記で述べた秘匿の安全性と改ざん検知の安全性の和で上界を取ることができる.

2.3　認証暗号を実現するための暗号プリミティブ

認証暗号を実現するための固定長入出力の暗号部品として Tweakable ブロック暗号 (Tweakable block cipher; 以下 TBC) が挙げられる. TBC はブロック暗号の拡張であり, 秘密鍵と平文の他に Tweak と呼ばれる公開調整値を入力に持つ, 固定長入出力の鍵付き置換である [82, 83]. TBC は暗号

化関数 $\widetilde{E}$ と復号関数 $\widetilde{D}$ の対で定義される. 暗号化関数の入力は秘密鍵 K, n ビット平文 M, t ビット Tweak T であり, 出力は n ビットの暗号文 C である. これを $\widetilde{E}_K^T(M) = C$ と表記する. 復号関数は秘密鍵 K, n ビット暗号文 C, t ビット Tweak T を入力として取り, n ビットの平文 M を出力する. これを $\widetilde{D}_K^T(C) = M$ と表し, $\widetilde{D}_K^T(\widetilde{E}_K^T(M)) = M$ である. ここで $t = 0$ の場合は, $\widetilde{E}_K^T$, $\widetilde{D}_K^T$ はブロック暗号と見なすことができる.

自然数 n に対して $\mathrm{Perm}(n)$ を n ビット入出力の置換全体の集合とする. また, $\tilde{\pi} : \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$ を, $\forall T \in \{0,1\}^t$ について $\tilde{\pi}(T, \cdot) \in \mathrm{Perm}(n)$ となる写像であるとする. ここで $\mathrm{TPerm}(t, n)$ を上記のような $\tilde{\pi}$ の全体集合とする. $\widetilde{\mathsf{P}} \xleftarrow{\$} \mathrm{TPerm}(n)$ なる $\widetilde{\mathsf{P}}$ を, Tweakable uniform random permutaion (TURP) と呼ぶ. TBC の安全性は TURP との判別成功確率で定義される. 選択平文攻撃を行う攻撃者に対して TURP との判別が困難である TBC を Tweakable pseudo random permutaion (TPRP) と呼ぶ. ブロック暗号の場合の安全性は $P \xleftarrow{\$} \mathrm{Perm}(n)$ なるランダム置換 P との判別成功確率で定義され, 選択平文攻撃を行う攻撃者に対してランダム置換との判別が困難であるブロック暗号を Pseudo random permutaion (PRP) と呼ぶ.

また, 認証暗号のプリミティブとして鍵なし置換が用いられる場合もある. その場合の認証暗号の安全性評価においては, 用いる置換を公開ランダム置換と置き換えて証明を行うのが一般的である.

第3章　Elephant

Elephant [21] は暗号学的置換をプリミティブとして用いた認証暗号である．用いる置換は ISO/IEC 29192-5:2016 [68] や FIPS 202 [5] において標準化されているものと同じもしくは同様である．モード構成については NIST LWC ファイナリストのうち唯一高度な並列実行が可能な方式であり, 他ファイナリストとは設計思想が大きく異なるのが特徴である．2022 年 9 月末時点で, 提案者らが仕様書において主張する安全性を損なう解析結果は発表されていない．

3.1　準備

自然数 n について, $X_1 \dots X_\ell \xleftarrow{n} X$ は X を $\ell = \lceil |X|/n \rceil$ 個の n ビットブロックに分割することを指す．但し $|X|/n \neq \lceil |X|/n \rceil$ のとき, 最後のブロック X_ℓ は n ビットになるよう 0 でパディングされている．ビット列 $X \in \{0,1\}^n$, 自然数 $i \leq n$ について, $X \ll i$ (resp. $X \gg i$) を X の i ビット左論理シフト (resp. 右論理シフト) とし, $X \lll i$ (resp. $X \ggg i$) を X の i ビット左巡回シフト (resp. 右巡回シフト) とする．ビット列 $X \in \{0,1\}^*$, 自然数 $i \leq |X|$ について, $\lfloor X \rfloor_i$ は X の左 i ビット列を指す．

3.2　仕様

Elephant は暗号学的置換をプリミティブとして用いた認証暗号利用モードの名称であり, かつ 3 つの認証暗号 Dumbo, Jumbo, Delirium をまとめた総称である．この 3 方式は全てモード構成が Elephant であり, 用いる置換がそれぞれ異なる．Dumbo, Jumbo, Delirium はそれぞれ Spongent-π[160], Spongent-π[176], Keccak-f[200] を暗号プリミティブとして用いる．つまり, Dumbo は Elephant-Spongent-π[160], Jumbo は Elephant-Spongent-π[176], Delirium は Elephant-Keccak-f[200] である．Primary member は Dumbo である．

3.2.1　モード

本節では Elephant のモード構成と 3 つのインスタンスの仕様について述べる．まずモードの概要として, 認証暗号モードとしての Elephant の構成は Enc-then-MAC 構造と呼ばれるもので, まず平文を暗号化したのち暗号文をメッセージ認証コード (Message authentication code; MAC) で処理してタグを導出する．暗号化部分は CTR モード, MAC 部分は Protected counter sum [14, 84] と同様の構成である．暗号化や MAC の内部では置換を用いた TBC 構成が使われており, その構成は Masked Even-Mansour [54] を簡易にしたものである．

より詳細な定義として, Elephant の暗号化関数アルゴリズムを Alg. 1 に, 図を図 3.1 に示す．但し P は n ビット入出力の置換である．復号関数は省略する．暗号化関数は入力として k ビット秘密鍵 K, m ビットのナンス N, 任意長の AD A, 任意長の平文 M を取り, 出力は $|M|$ ビット暗号文 C, t ビットタグ T である．具体的なパラメータや用いる置換等は各インスタンスにより異なる．その違いは表 3.1 の通りである．また, 関数 mask について φ_1: $\{0,1\}^n \to \{0,1\}^n$ を線形帰還シフトレジスタ

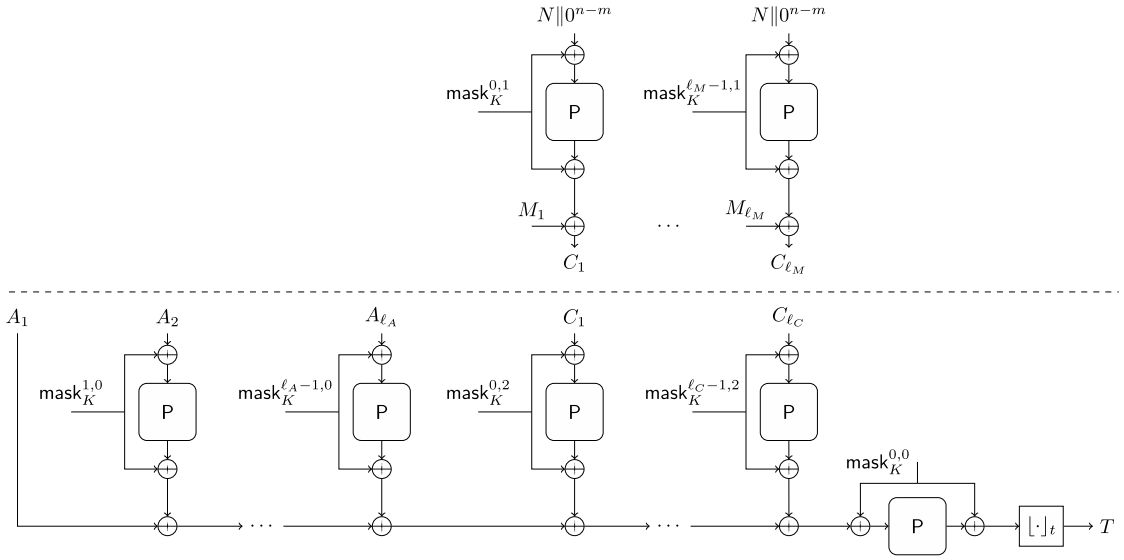


図 3.1: Elephant の暗号化関数. Elephant のウェブサイト [3] より引用.

表 3.1: Elephant の各インスタンスにおける違い. 文献 [21] より一部引用.

方式	鍵長 k	ナンス長 m	置換サイズ n	タグ長 t	置換 P	φ_1
Dumbo	128	96	160	64	Spongent-π[160]	(3.1)
Jumbo	128	96	176	64	Spongent-π[176]	(3.2)
Delirium	128	96	200	128	Keccak-f[200]	(3.3)

(Linear feedback shift register; LFSR), id を恒等関数とし, $\varphi_2 = \varphi_1 \oplus \mathsf{id}$ と定義する. このとき, 関数 mask は以下の通り定義される.

$$\mathsf{mask}_K^{a,b} = \varphi_2^b \circ \varphi_1^a \circ \mathsf{P}(K \,\|\, 0^{n-k}).$$

φ_1 の定義はインスタンスにより異なる. $0 \le i \le 24$ について, x_i を 8 ビットワードとするとき, Dumbo, Jumbo, Delirium における φ_1 はそれぞれ以下の通り定義される.

$$(x_0, \ldots, x_{19}) \mapsto (x_1, \ldots, x_{19}, x_0 \lll 3 \oplus x_3 \ll 7 \oplus x_{13} \gg 7), \tag{3.1}$$

$$(x_0, \ldots, x_{21}) \mapsto (x_1, \ldots, x_{21}, x_0 \lll 1 \oplus x_3 \ll 7 \oplus x_{19} \gg 7), \tag{3.2}$$

$$(x_0, \ldots, x_{24}) \mapsto (x_1, \ldots, x_{24}, x_0 \lll 1 \oplus x_2 \lll 1 \oplus x_{13} \ll 1). \tag{3.3}$$

3.2.2 プリミティブ

Elephant で用いられている暗号プリミティブはインスタンスにより異なる. 本節では, 用いられている 3 つの置換についてそれぞれ概要を説明したのち, Primary member の Dumbo で使用されている Spongent-π[160] の仕様を説明する.

概要 Spongent-π[160] は 160 ビット入出力の置換である. そのラウンド関数は, CHES 2011 で Bogdanov らによって提案されたハッシュ関数 Spongent [24] で用いられている内部置換のラウンド関数を基に Elephant 提案者らによって設計されている [21]. なお, Spongent は ISO/IEC 29192-5:2016 [68] において標準化されている. またラウンド数は, 文献 [24] にて用いられている Spongent 内部置換のラウンド数決定方法に則って 80 段と決定されている.

Algorithm 1 Elephant encryption algorithm enc

Input key $(K, N, A, M) \in \{0,1\}^k \times \{0,1\}^m \times \{0,1\}^* \times \{0,1\}^*$
Output $(C, T) \in \{0,1\}^{|M|} \times \{0,1\}^t$

$M_1 \dots M_{\ell_M} \xleftarrow{n} M$
for $i = 1, \dots, \ell_M$ **do**
  $C_i \leftarrow M_i \oplus \mathsf{P}(N \,\|\, 0^{n-m} \oplus \mathsf{mask}_K^{i-1,1}) \oplus \mathsf{mask}_K^{i-1,1}$
$C \leftarrow \lfloor C_1 \dots C_{\ell_M} \rfloor_{|M|}$
$A_1 \dots A_{\ell_A} \xleftarrow{n} N \,\|\, A \,\|\, 1$
$C_1 \dots C_{\ell_C} \xleftarrow{n} C \,\|\, 1$
$T \leftarrow A_1$
for $i = 2, \dots, \ell_A$ **do**
  $T \leftarrow T \oplus \mathsf{P}(A_i \oplus \mathsf{mask}_K^{i-1,0}) \oplus \mathsf{mask}_K^{i-1,0}$
for $i = 1, \dots, \ell_C$ **do**
  $T \leftarrow T \oplus \mathsf{P}(C_i \oplus \mathsf{mask}_K^{i-1,2}) \oplus \mathsf{mask}_K^{i-1,2}$
$T \leftarrow \mathsf{P}(T \oplus \mathsf{mask}_K^{0,0}) \oplus \mathsf{mask}_K^{0,0}$
return $(C, \lfloor T \rfloor_t)$

Spongent-π[176] は 176 ビット入出力の置換である．Spongent-π[176] はハッシュ関数 Spongent のインスタンス Spongent-160 [1]の内部で用いられている 176 ビット入出力の内部置換と同じであり，ラウンド数は 90 段である．

Keccak-f[200] は 200 ビット入出力の置換である．Keccak-f[200] はハッシュ関数 Keccak [17] の 200 ビット入出力内部置換として定義されているものと同じであり，ラウンド数は 18 段である．なお，Keccak ハッシュは 1600 ビット入出力置換を用いたインスタンスが FIPS 202 [5] において標準化されている．

Spongent-π[160] の仕様　160 ビットの入力 X に対し，Spongent-π[160] の出力は以下のように定義される．

$$
\begin{aligned}
&\text{for } i = 1, \dots, 80 \text{ do} \\
&\quad X \leftarrow X \oplus 0^{153} \,\|\, \mathsf{ICounter}_{160}(i) \oplus \mathsf{rev}(0^{153} \,\|\, \mathsf{ICounter}_{160}(i)) \\
&\quad X \leftarrow \mathsf{sBoxLayer}_{160}(X) \\
&\quad X \leftarrow \mathsf{pLayer}_{160}(X)
\end{aligned}
$$

ここで，rev は入力のビット順番を逆にする操作である．$\mathsf{ICounter}_{160}$ は 7 ビット LFSR で，帰還多項式が $x^7 + x^6 + 1$ であり，初期値が 1110101 のものである．$\mathsf{sBoxLayer}_{160}$ は表 3.2 で定義される 4 ビット S-box を 40 個並列に並べたものである．pLayer_{160} は入力の j 番目のビットを以下で定義する $P_{160}(j)$ 番目のビットに移動させる操作である．

$$
P_{160}(j) = \begin{cases} 40 \cdot j \bmod 159 & (j \in \{0, \dots, 158\}) \\ 159 & (j = 159) \end{cases}
$$

[1]ハッシュ関数 Spongent-160 はハッシュ長が 160 ビットであることを指す．Spongent-π[160] との混同に注意されたい．

表 3.2: Spongent-π[160] の $\mathsf{sBoxLayer}_{160}$ 内部で用いる 4 ビット S-box. 16 進数表現で記載.

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(X)$	E	D	B	0	2	1	4	F	7	A	8	5	9	C	3	6

3.3 安全性

2022 年 9 月現在, Elephant はいずれの方式についても仕様書で主張される安全性を脅かす攻撃は提案されていない. 本節では, まず 3.3.1 節で提案者らが主張する安全性を概説し, 3.3.2 節でモードの安全性の根拠及び第三者評価について述べる. 3.3.3 節にて用いる置換及び関連方式の解析状況, 3.3.4 節にてその他の解析について述べる.

3.3.1 仕様上の安全性

表 3.3: Elephant のインスタンス 3 方式に対する安全性 [3, 21].

方式	解読・改ざんに必要な オフライン計算量	オンライン計算量の上限
Dumbo	2^{112}	2^{50} バイト
Jumbo	2^{127}	2^{50} バイト
Delirium	2^{127}	2^{74} バイト

提案者らが主張する安全性を表 3.3 に示す. ここで表内の「オンライン計算量の上限」とは, 1 つの鍵で処理可能なデータ量の上限を指す. また「解読・改ざんに必要なオフライン計算量」とは, 公開ランダム置換へのアクセス回数を指す. これは, 攻撃者に対してオンライン計算量の上限まで暗号化オラクルと復号オラクルへのアクセスを許した場合に, 暗号文解読やタグ改ざんのために攻撃者が必要となる時間計算量に相当する. 表 3.3 で示す安全性は, 提案者らによって与えられている Elephant モードの安全性証明 [21] から導出される. その安全性証明では, 2.2 節にて述べた秘匿と改ざん検知の安全性を合わせた安全性定義 [95] において用いる置換を公開ランダム置換としたときに, Nonce respecting の下で Elephant モードの安全性が示されている.

3.3.2 モードの安全性

文献 [21, 22] において Elephant モードの安全性証明が示されている. その安全性証明は, Enc-then-MAC 構造については Bellare と Namprempre による証明 [12] 及び Namprempre らによる証明 [92], MAC 部分については Bernstein による証明 [14] 及び Luykx らによる証明 [84], 置換ベース TBC については Granger らによる証明 [54] のアイディアがそれぞれベースとなっている. 加えて, マルチユーザーセキュリティの安全性証明も文献 [22] にて示されており, シングルユーザーで示されていた表 3.3 の安全性がマルチユーザーの場合でも成立することが示されている. また提案者らは, Nonce misuse 設定の場合には秘匿の安全性は担保できないものの, 改ざん検知についての安全性は保たれると述べている [21]. 加えて, 未検証平文が攻撃者に与えられる設定 (Release of unverified plaintext [7]) の場合の安全性も担保できると述べている [21].

Elephant モードへの第三者評価としては, 土生と岩田による Elephant モードに対する鍵回復, 識別及び偽造攻撃 [117] が挙げられる. これらの攻撃に必要な計算量が, Elephant モードの安全性証明により示される安全性バウンドが要求する攻撃成功に必要な計算量と同等であるため, そのバウンド

がタイトであることを示す結果である．よって仕様書で主張される安全性を脅かすものではない．他にも, Elephant やその内部で用いるモード構成の耐量子安全性に関して攻撃や証明がいくつか発表されている [6, 26, 98] が, 提案者らは Elephant の耐量子安全性については特に主張していないため, 仕様上の安全性とは矛盾しない．

3.3.3 プリミティブ及び関連方式の安全性

本節では, Elephant で用いられている 3 つの置換 Spongent-π[160], Spongent-π[176], Keccak-f[200] に対する安全性について述べる．

Spongent-π[160] と Spongent-π[176] 3.2.2 節にて述べた通り, Spongent-π[160] は厳密には Elephant 提案者らによって定義された置換である．そのため本節ではまず, Spongent-π[160] のラウンド数決定方法について述べたのち, Spongent-π[160] 及び Spongent-π[176] の第三者評価について述べる．

ハッシュ関数 Spongent の内部置換に対して, 次の定理が知られている．

定理 3.3.1 (文献 [24] の Theorem 1, 文献 [21] の Theorem 5.1.) $b \geq 64$ に対し, Spongent-$\pi[b]$ をブロックサイズ b の Spongent 内部置換とする．このとき, いかなる差分に対しても 5 ラウンドの Spongent-$\pi[b]$ には少なくとも 10 個の active S-box が存在する．

Bogdanov らは Spongent 提案論文 [24] にて, この定理及び Spongent 内部置換で用いている S-box の最大差分確率が 2^{-2} であることを用いて, (安全性マージンを含めて) 少なくとも b 個の active S-box が存在するように Spongent 内部置換のラウンド数を決定した．また Bogdanov らは同論文において, そのように決めたラウンド数の Spongent 内部置換が線形解析に対しても安全性を持つことを述べている．Elephant 提案者らもこのラウンド決定方法に従って Spongent-π[160] のラウンド数を決定しており, 同様に Spongent-π[160] が線形解析に対しても十分な安全性を持つことを示している [21]．

次に, Spongent-π[160] と Spongent-π[176] に対する第三者評価について述べる． 伊藤による調査 [116] によると, 2021 年 9 月時点では Zhang と Liu による中間一致手法を用いた切り詰め差分攻撃 [110] が Spongent 内部置換に対する最良の攻撃であった．文献 [110] では Spongent-π[176] に対して中間一致攻撃のラウンド数が 7 段, 切り詰め差分攻撃のラウンド数が 46 段であり, 識別攻撃可能段数が合わせて 53 段であることが示されている．これ以降の解析として, Sun らによる 21 段 Spongent-π[176] に対するゼロサム識別器の提案[2] [102] と, 後ほど述べる Schrottenloher と Stevens による中間一致攻撃 [97] が挙げられるが, 現時点では Spongent-π[160], Spongent-π[176] 共に致命的な脆弱性は報告されていない．Spongent-π[176] に対する現在最良の識別攻撃は, 文献 [110] の攻撃と Schrottenloher と Stevens による攻撃 [97] の組み合わせである．Spongent-π[176] に対して, 文献 [110] における中間一致攻撃のラウンド数が文献 [97] によって 2 段改良されたことにより, Spongent-π[176] の識別攻撃可能段数は現在 55 段である．文献 [97] では Spongent-π[160] に対する中間一致攻撃のラウンド数が 9 段であることも示されているが, 文献 [110] では Spongent-π[160] への切り詰め差分攻撃可能段数が示されていないため, 中間一致手法を用いた切り詰め差分攻撃による Spongent-π[160] の識別攻撃可能段数は不明である．

Keccak-f[200] Keccak-f[200] に対する解析状況として以下 3 つについて述べる．

- **Keccak-f[200] に対する解析.** Keccak-f[200] への識別攻撃について表 3.4 にまとめる．表中の 2 つの解析結果はそれぞれ攻撃者の設定が異なるため, 直接的な比較は困難であることに注

[2] この結果は文献 [21] において Spongent-π[160] に対する解析として引用されているが, それは間違いである．Sun らによる解析では Spongent-160 を 160 ビット出力のハッシュとして表記しており, その内部置換の解析を行っているため, 正しくは Spongent-π[176] に対する解析となる．

表 3.4: Keccak-$f[200]$ に対する識別攻撃.

Type	Target	Rounds	Time	Method	Reference
Distinguisher	Permutation	7 / 18	$2^{46.0}$	Limited-Birthday	[50]
	Permutation	6 / 18	$2^{142.0}$	Differential	[87]

表 3.5: Keccak-$f[200]$ を内部置換として用いた Keccak ハッシュへの解析. Keccak$[r, c]$ はスポンジ構造のレートが r ビット, キャパシティが c ビットであることを示す. 文献 [116] 表 5.1 より一部引用.

Instance	Rounds	Attack type	Time	Reference
Keccak$[40, 160]$	2	Algebraic collision	$2^{73.0}$	[25]
Keccak$[72, 128]$	2	Algebraic collision	$2^{52.5}$	[25]

意されたい. また, 表では明示的に Keccak-$f[200]$ への解析結果を示した文献のみを挙げているが, 異なる入出力サイズの Keccak-f 置換に対する有効な解析手法としてゼロサム攻撃 [108] 等が知られている. それらが Keccak-$f[200]$ に対して適用可能であるかどうかは本報告書では検証していない.

- **Keccak-f[200] を用いた Keccak ハッシュに対する解析.** Keccak [17] はスポンジ構造のハッシュ関数であり, Keccak-$f[200]$ を内部置換として用いるインスタンスが存在する. 伊藤による調査 [116] によると, 2021 年 9 月時点では 2 段の Keccak-$f[200]$ を用いた Keccak に対する Boissier らによる代数衝突攻撃 [25] が最良の攻撃であり, 現時点においてもこれが最良である.

- **認証暗号 Ketje Jr v1, Ketje Jr v2 に対する解析.** Ketje Jr v1, Ketje Jr v2 はスポンジ構造の認証暗号 Ketje [18] のインスタンスである. なお, Ketje は認証暗号コンペティション CAESAR [2] の第三ラウンド候補である. Ketje Jr v1, Ketje Jr v2 の内部置換として Keccak-$f[200]$ のバリアントでラウンド数 12 段のものが用いられており, レートは 16 ビットである. 伊藤による調査 [116] でまとめられている解析結果 [48, 52, 99, 112] と, それに含まれないいくつかの解析結果 [109, 113] を表 3.5 にまとめる. 鍵回復攻撃については, ラウンド数が 5 段及び 6 段までの (条件付き) キューブ攻撃が提案されており, ステート回復攻撃については 12 段の分割統治攻撃が提案されている.

Delirium で用いられる Keccak-$f[200]$ は 18 ラウンドであることや, スポンジ構造のレートに当たる部分が Elephant には存在しない, つまり攻撃者が置換の入出力を直接計算できる部分が Elephant にはないことから, これらの攻撃は Keccak-$f[200]$ 及び Delirium の安全性を脅かすものではない. しかしながら, Keccak ハッシュや認証暗号 Ketje は内部置換として Keccak-$f[200]$ と同様の構造を用いているため, 今後も安全性評価の上では考慮に入れる必要がある.

3.3.4 その他

その他の解析としては, Zhou らによるラウンド数を 8 段に削減した Keccak-$f[200]$ を用いた Delirium への補間攻撃 [114] が挙げられる. 攻撃に必要な時間計算量は $2^{98.3}$ XOR 相当, 空間計算量は 2^{70} ビット, 必要なデータ量は 2^{70} ブロックである. Delirium では 18 段の Keccak-$f[200]$ が用いられているため, 仕様書において主張される安全性を脅かすものではない.

また, サイドチャネル攻撃等の非ブラックボックス安全性に関する解析として, Vialar によるサイドチャネル攻撃を用いた Dumbo 及び Jumbo の鍵回復攻撃 [106], Takemoto らによるハードウェアトロイが埋め込まれた Elephant 実装に対する安全性評価 [103], Joshi と Mazumdar によるフォールト攻撃を用いた Dumbo の鍵回復攻撃 [74], Madushan らによるフォールト攻撃を用いた Dumbo の鍵回

表 3.6: Ketje Jr への解析. 文献 [116] 表 7.5 より一部引用.

Instance	Key size	Rounds	Attack type	Method	Time	Reference
Ketje Jr v1	72	6	Key recovery	Cube	$2^{68.0}$	[99]
Ketje Jr v1	96	5	Key recovery	Cube	$2^{56.0}$	[48]
Ketje Jr v1	96	5	Key recovery	Cube	$2^{36.9}$	[99]
Ketje Jr v1	96	5	Key recovery	Conditional cube	$2^{26.6}$	[113]
Ketje Jr v1	96	12(full)	State recovery (rate: 16)	Divide and conquer	$2^{152.0}$	[52]
Ketje Jr v1	96	12(full)	State recovery (rate: 24)	Divide and conquer	$2^{128.0}$	[52]
Ketje Jr v1	96	12(full)	State recovery (rate: 32)	Divide and conquer	$2^{92.0}$	[52]
Ketje Jr v1	96	12(full)	State recovery (rate: 32)	Divide and conquer	$2^{87.6}$	[109]
Ketje Jr v1	96	12(full)	State recovery (rate: 40)	Divide and conquer	$2^{71.9}$	[52]
Ketje Jr v1	96	12(full)	State recovery (rate: 40)	Divide and conquer	$2^{71.6}$	[109]
Ketje Jr v2	80	6	Key recovery	Cube	$2^{59.2}$	[99]
Ketje Jr v2	96	5	Key recovery	Cube	$2^{50.3}$	[48]
Ketje Jr v2	96	5	Key recovery	Cube	$2^{34.9}$	[99]
Ketje Jr v2	96	5	Key recovery	Cube	$2^{30.5}$	[112]
Ketje Jr v2	96	5	Key recovery	Conditional cube	$2^{27.5}$	[113]
Ketje Jr v2	96	12(full)	State recovery (rate: 16)	Divide and conquer	$2^{152.0}$	[52]
Ketje Jr v2	96	12(full)	State recovery (rate: 24)	Divide and conquer	$2^{128.0}$	[52]
Ketje Jr v2	96	12(full)	State recovery (rate: 32)	Divide and conquer	$2^{104.0}$	[52]
Ketje Jr v2	96	12(full)	State recovery (rate: 40)	Divide and conquer	$2^{82.3}$	[52]

復攻撃 [85] が挙げられる. 提案者らは Elephant の非ブラックボックス安全性については特に主張していないため, これらの解析も仕様書で主張される安全性とは矛盾しない. 提案者らは, ユーザーがこれらの攻撃を防ぎたい場合は実装レベルでの適切な対策が必要であると文献 [21] にて述べている.

第4章　ISAP

Isap [38] は暗号学的置換をプリミティブとして用いた方式である. 用いる置換の一部は FIPS 202 [5] において標準化されているものと同じである. モードは他の多くの NIST LWC ファイナリストと同じくスポンジ構造 [16] を採用しているが, Fresh rekeying [86] から着想を得てサイドチャネル攻撃に対して堅牢となるような設計となっているのが特徴である. 2022 年 9 月末時点で, 提案者らが仕様書において主張する安全性を損なう解析結果は発表されていない.

4.1　準備

ビット列 $X \in \{0,1\}^*$ と自然数 $n \leq |X|$ に対し, $\lceil X \rceil^n$ を X の上位 n ビットのビット列, $\lfloor X \rfloor_n$ を X の下位 n ビットのビット列を表す. スポンジ構造の n ビットステート S に対し, S_r は S の r ビット外部パート (つまりレート部分), S_c は S の c ビット内部パート (つまりキャパシティ部分) を指す.

4.2　仕様

Isap は暗号学的置換をプリミティブとして用いた認証暗号利用モードの名称であり, かつ 4 つの認証暗号 Isap-A-128a, Isap-K-128a, Isap-A-128, Isap-K-128 をまとめた総称である. この 4 方式は全てモード構成が Isap であり, 用いる置換やパラメータがそれぞれ異なる. Isap-A-128a, Isap-A-128 はプリミティブとして Ascon-p, Isap-K-128a, Isap-K-128 はプリミティブとして Keccak-p[400] を用いる. Primary member は Isap-A-128a である.

加えて提案者らは Isap のハッシュ関数として, 既存のハッシュ関数で, かつ各インスタンスと同じ置換を用いる方式を推奨している. 詳細には, Isap-A-128a 又は Isap-A-128 を用いる場合は NIST LWC ファイナリスト Ascon [41] のハッシュ関数 AsconHash を, Isap-K-128a, Isap-K-128 を用いる場合は NIST FIPS 202 [5] で定められる sponge[Keccak-p[400, 16], pad10*1, 144]$(M \,\|\, 01, 256)$ 及び sponge[Keccak-p[400, 20], pad10*1, 144]$(M \,\|\, 01, 256)$ をそれぞれ用いることを推奨している.

4.2.1　モード

本節では Isap のモード構成と 4 つのインスタンスの仕様について述べる. Isap は FSE 2017 で提案された同名の方式 [40] をベースに作られている. まずモードの概要として, 認証暗号全体の構成は Elephant と同じく Enc-then-MAC 構造であるが, Isap にはそれに加えて Rekey と呼ばれる内部関数が存在する. Rekey 関数は, Isap 内部の暗号化部分とメッセージ認証コード (MAC) 部分で用いる鍵を, (マスター) 秘密鍵とナンスから導出する関数である. Rekey 関数, 内部の暗号化, MAC 部分はそれぞれ IsapRk, IsapEnc, IsapMac と呼ばれ, その全てがスポンジ構造を用いて構成されている.

より詳細な定義として, Isap の暗号化アルゴリズムを Alg. 2 に, 内部関数 IsapRk, IsapEnc, IsapMac の図をそれぞれ図 4.1a, 4.1b, 4.1c に示す. 復号関数は省略する. 暗号化関数は入力として k ビット秘密鍵 K, k ビットナンス N, 任意長 AD A, 任意長平文 M を取り, 出力は $|M|$ ビット暗号文 C, k ビットタグ T である. 具体的なパラメータや用いる置換等は各インスタンスにより異なる.

Algorithm 2 $\mathcal{E}(K, N, A, M)$

Input key $K \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$,
associated data $A \in \{0,1\}^*$,
plaintext $M \in \{0,1\}^*$
Output ciphertext $C \in \{0,1\}^{|M|}$, tag $T \in \{0,1\}^k$

$C \leftarrow \textsc{IsapEnc}(K, N, M)$
$T \leftarrow \textsc{IsapMac}(K, N, A, C)$
return C, T

Algorithm 3 $\textsc{IsapEnc}(K, N, M)$

Input key $K \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$,
plaintext $M \in \{0,1\}^*$
Output ciphertext $C \in \{0,1\}^{|M|}$

$M_1 \dots M_t \leftarrow r_{\text{H}}$-bit blocks of $M \,\|\, 0^{-|M| \bmod r_{\text{H}}}$
$K^*_{\text{E}} \leftarrow \textsc{IsapRk}(K, \text{ENC}, N)$, $S \leftarrow K^*_{\text{E}} \,\|\, N$
for $i = 1, \dots, t$ **do**
  $S \leftarrow p_{\text{E}}(S)$, $C_i \leftarrow S_{r_{\text{H}}} \oplus M_i$
$C \leftarrow \lceil C_1 \,\|\, \dots \,\|\, C_t \rceil^{|M|}$
return C

Algorithm 4 $\textsc{IsapRk}(K, f, Y)$

Input key $K \in \{0,1\}^k$, flag $f \in \{\text{ENC}, \text{MAC}\}$,
string $N \in \{0,1\}^k$
Output session key $K^* \in \{0,1\}^z$

if $f = \text{ENC}$ **then**
  $(\text{IV}, z) \leftarrow (\text{IV}_{\text{KE}}, n - k)$
else
  $(\text{IV}, z) \leftarrow (\text{IV}_{\text{KA}}, k)$
$Y_1 \dots Y_w \leftarrow r_{\text{B}}$-bit blocks of $Y \,\|\, 0^{-k \bmod r_{\text{B}}}$
$S \leftarrow K \,\|\, \text{IV}$
$S \leftarrow p_{\text{K}}(S)$
for $i = 1, \dots, w-1$ **do**
  $S \leftarrow p_{\text{B}}((S_{r_{\text{B}}} \oplus Y_i) \,\|\, S_{c_{\text{B}}})$
$S \leftarrow p_{\text{K}}((S_{r_{\text{B}}} \oplus Y_w) \,\|\, S_{c_{\text{B}}})$
$K^* \leftarrow \lceil S \rceil^z$
return K^*

Algorithm 5 $\textsc{IsapMac}(K, N, A, C)$

Input key $K \in \{0,1\}^k$, nonce $N \in \{0,1\}^k$,
associated data $A \in \{0,1\}^*$, ciphertext $C \in \{0,1\}^*$
Output tag $T \in \{0,1\}^k$

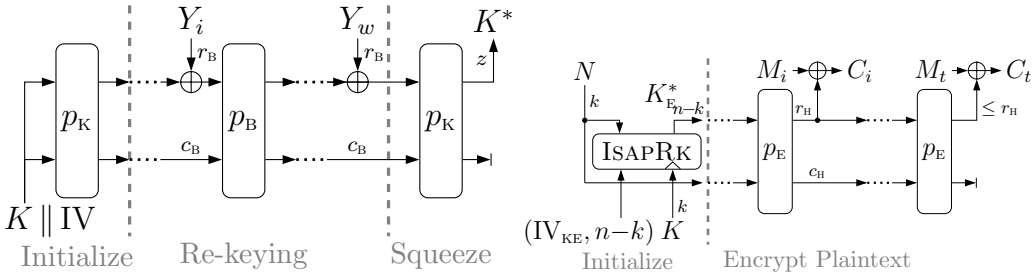
$A_1 \dots A_s \leftarrow r_{\text{H}}$-bit blocks of $A \,\|\, 1 \,\|\, 0^{-|A|-1 \bmod r_{\text{H}}}$
$C_1 \dots C_t \leftarrow r_{\text{H}}$-bit blocks of $C \,\|\, 1 \,\|\, 0^{-|C|-1 \bmod r_{\text{H}}}$
$S \leftarrow N \,\|\, \text{IV}_{\text{A}}$, $S \leftarrow p_{\text{H}}(S)$
for $i = 1, \dots, s$ **do**
  $S \leftarrow p_{\text{H}}((S_{r_{\text{H}}} \oplus A_i) \,\|\, S_{c_{\text{H}}})$
$S \leftarrow S \oplus (0^{n-1} \,\|\, 1)$
for $i = 1, \dots, t$ **do**
  $S \leftarrow p_{\text{H}}((S_{r_{\text{H}}} \oplus C_i) \,\|\, S_{c_{\text{H}}})$
$K^*_{\text{A}} \leftarrow \textsc{IsapRk}(K, \text{MAC}, \lceil S \rceil^k)$
$S \leftarrow p_{\text{H}}(K^*_{\text{A}} \,\|\, \lfloor S \rfloor_{n-k})$
$C \leftarrow \lceil C_1 \,\|\, \dots \,\|\, C_t \rceil^{|M|}$, $T \leftarrow \lceil S \rceil^k$
return T

その違いは表 4.1 の通りである．また, 内部で用いる初期ベクトルも各インスタンス等により異なる．その違いは表 4.2 の通りである．

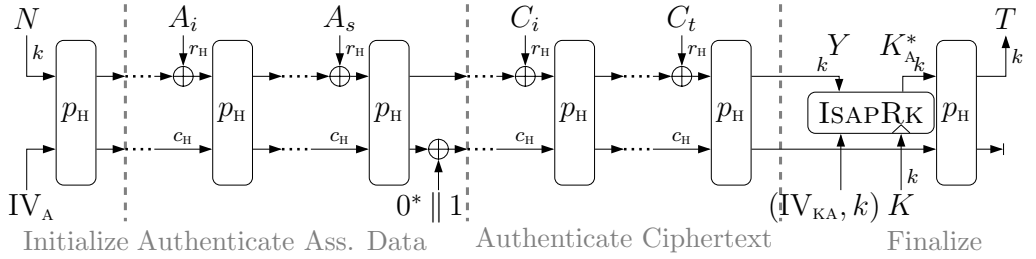
耐漏洩安全性のための実装　Isap はサイドチャネル攻撃に対して堅牢となるように設計されているが, その耐漏洩安全性を達成するための実装上の要件が存在する．その詳細は文献 [13, 38, 105] を参照されたいが, 概要としては Fresh rekeying [86] のアイディアと同じく, マスター秘密鍵から一時鍵を導出する IsapRk は Differential power analysis (DPA) 攻撃に対して安全, 一時鍵が入力される IsapEnc と IsapMac は Simple power analysis (SPA) 攻撃に対して安全となるような実装が求められる．

4.2.2　プリミティブ

Isap で用いられている暗号プリミティブはインスタンスにより異なる．Ascon-p は 320 ビット入出力の置換であり, そのラウンド関数は NIST LWC ファイナリストの Ascon [41] で用いられている置換と同じである．その詳細は Ascon に対する安全性評価レポートを参照されたい．ラウンド数は各インスタンスや内部プロセスにより異なり, 表 4.1 の通りである．Ascon は NIST LWC 前に開かれた認証暗号コンペティション CAESAR [2] においても Lightweight applications のカテゴリで最終ポートフォリオに選出されている．Keccak-p[400] は 400 ビット入出力の置換であり, そのラウンド関数は FIPS 202 [5] で定められている置換と同じである．ラウンド数は各インスタンスや内部プロセスにより異なり, 表 4.1 の通りである．なお, Elephant で用いられている Keccak-f と Keccak-p



(a) IsapRk. **(b)** IsapEnc.



(c) IsapMac.

図 4.1: Isap の内部関数 3 つの図. Isap のウェブサイト [4] より引用. 厳密には, 図 4.1b と図 4.1c の IsapRk にはそれぞれ $(\mathrm{IV}_{KE}, n-k)$, (IV_{KA}, k) でなく flag f が入力される.

表 4.1: Isap の各インスタンスで用いる置換及びパラメータ. 文献 [38] より引用. ここで, Permutation の列は Isap 各インスタンスで用いる置換 p_H, p_B, p_E, p_K の種類を指し, p_H, p_B, p_E, p_K のラウンド数はそれぞれ s_H, s_B, s_E, s_K で表される.

Name	Permutation	Security level	Bit size of			Rounds			
		k	n	r_H	r_B	s_H	s_B	s_E	s_K
Isap-A-128a	Ascon-p	128	320	64	1	12	1	6	12
Isap-K-128a	Keccak-p[400]	128	400	144	1	16	1	8	8
Isap-A-128	Ascon-p	128	320	64	1	12	12	12	12
Isap-K-128	Keccak-p[400]	128	400	144	1	20	12	12	12

との違いはラウンド数が仕様上で明示的に定められているかどうかのみであり, ラウンド関数は同じである. Keccak-f は固定されたラウンド数を持つが, Keccak-p はラウンド数を変数として持つ.

4.3 安全性

2022 年 9 月末時点において, Isap はいずれの方式についても仕様書で主張される安全性を脅かす攻撃は提案されていない. 本節では, まず 4.3.1 節で提案者らが主張する安全性を概説し, 4.3.2 節でモードのブラックボックス安全性と耐漏洩安全性の 2 つについて, 根拠及び第三者評価について述べる. 4.3.3 節にて用いる置換の現時点での攻撃可能段数と, 同じ置換を用いる他の関連方式の攻撃可能段数について述べる. 4.3.4 節にてその他の解析について述べる.

4.3.1 仕様上の安全性

提案者らが主張する安全性レベルを表 4.3 に示す. 提案者らはこの安全性のために, 暗号化関数では同じナンスを使ってはならないこと (Nonce respecting) 及び, 復号関数における復号プロセスは,

表 4.2: Isap の各インスタンスで用いる初期ベクトル. 16 進数表現で記載. [38] より引用.

インスタンス	$\mathrm{IV_A}$	$1 \Vert k \Vert r_\mathrm{H} \Vert r_\mathrm{B} \Vert s_\mathrm{H} \Vert s_\mathrm{B} \Vert s_\mathrm{E} \Vert s_\mathrm{K} \Vert 0^*$
	$\mathrm{IV_{KA}}$	$2 \Vert k \Vert r_\mathrm{H} \Vert r_\mathrm{B} \Vert s_\mathrm{H} \Vert s_\mathrm{B} \Vert s_\mathrm{E} \Vert s_\mathrm{K} \Vert 0^*$
	$\mathrm{IV_{KE}}$	$3 \Vert k \Vert r_\mathrm{H} \Vert r_\mathrm{B} \Vert s_\mathrm{H} \Vert s_\mathrm{B} \Vert s_\mathrm{E} \Vert s_\mathrm{K} \Vert 0^*$
Isap-A-128a	$\mathrm{IV_A}$	01 80 4001 0C01060C 00*
	$\mathrm{IV_{KA}}$	02 80 4001 0C01060C 00*
	$\mathrm{IV_{KE}}$	03 80 4001 0C01060C 00*
Isap-K-128a	$\mathrm{IV_A}$	01 80 9001 10010808 00*
	$\mathrm{IV_{KA}}$	02 80 9001 10010808 00*
	$\mathrm{IV_{KE}}$	03 80 9001 10010808 00*
Isap-A-128	$\mathrm{IV_A}$	01 80 4001 0C0C0C0C 00*
	$\mathrm{IV_{KA}}$	02 80 4001 0C0C0C0C 00*
	$\mathrm{IV_{KE}}$	03 80 4001 0C0C0C0C 00*
Isap-K-128	$\mathrm{IV_A}$	01 80 9001 140C0C0C 00*
	$\mathrm{IV_{KA}}$	02 80 9001 140C0C0C 00*
	$\mathrm{IV_{KE}}$	03 80 9001 140C0C0C 00*

表 4.3: Isap 各インスタンスの安全性レベル. [38] より引用.

Requirement	Security in bits			
	Isap-A-128a	Isap-K-128a	Isap-A-128	Isap-K-128
Confidentiality of plaintext	128	128	128	128
Integrity of plaintext	128	128	128	128
Integrity of associated data	128	128	128	128
Integrity of nonce	128	128	128	128

タグ検証が成功してからでないと動いてはいけない仮定が必要であると述べている [38]. 文献 [38] では安全性定義が明示されてはいないものの, 以下の記述があることから表 4.3 の安全性は 2.2 節で述べたブラックボックス (つまりサイドチャネル攻撃を考慮しない) の安全性定義に基づくものであると推測する.

> All Isap family members provide 128-bit security against cryptographic attacks in the notion of nonce-based authenticated encryption with associated data (AEAD)

また文献 [38] の 5.1 節 Security of the Mode や文献 [39] の 2 章 Proofs の記述等から, 表 4.3 の安全性レベルはスポンジ構造に関するいくつかの安全性証明が根拠になっていると考えられる. これらの安全性証明はブラックボックスモデルであり, かつ用いる置換を公開ランダム置換と見なした場合に与えられるものである. しかしながら, 提案者らは文献 [38] で以下のように述べている.

> We emphasize that we do not require ideal properties for the permutations p_H, p_B, p_E, p_K. Non-random properties, including but not limited to inside-out zero sum distinguishers, of the permutations p_H, p_E, p_K, and of course p_B are known and do not automatically afflict the claimed security properties of the entire encryption algorithm.

文献 [39] でも同様の記述が複数見られる. つまり, Isap の安全性は一般的なスポンジ構造の証明可能安全性を 1 つの根拠にしてはいるものの, その証明可能安全性と実際の仕様に対する安全性には解

離があることに注意したい. このことから, ISAP の安全性を解析するためには, 用いる置換単体の解析や, 公開ランダム置換を仮定したモード単体の解析だけでなく, ISAP 全体の仕様を考慮した解析が必要不可欠である.

提案者らは加えて, ISAP に対する耐漏洩安全性 (Leakage-resilient security) の証明も与えている. この証明もまた, 用いる置換を公開ランダム置換と見なした場合に与えられるものである. 詳細は 4.3.2 節にて述べる.

4.3.2 モードの安全性

本節では, ISAP モードの証明可能安全性についてブラックボックス安全性に関するものと耐漏洩安全性に関するものについて述べる. つまり本節において, 用いる置換は全て公開ランダム置換であるという仮定を置く.

ブラックボックス安全性 ISAPRK 及び ISAPENC の安全性は, Daemen らによる Keyed duplex の証明可能安全性 [33] に依拠していることが提案者らによって述べられている [38] が, 厳密な安全性証明が与えられている訳ではない. 筆者個人の意見としては, 主張される 128 ビット安全性は Bertoni らによるスポンジ構造への安全性証明で, キャパシティ長の半分が安全性レベルとなるという結果 [20] に依拠しているのではないかと推測する. ISAPMAC に関しては Dobraunig と Mennink [44] によって Suffix keyed sponge の安全性証明が与えられており, ここから ISAPMAC の 128 ビット安全性が導出される. 但し, ISAPMAC の安全性バウンドを導出する際には, ISAPRK にあたる部分が特定の性質 (出力の一様性と衝突困難性) を満たす部品として理想化されていることに注意されたい. また, 同文献で示される Suffix keyed sponge の安全性バウンドがタイトであることが, 後に Dobraunig と Mennink によって示されている [45]. ハッシュ関数の安全性に関しては, スポンジ構造がランダムオラクルと Indifferentiability を持ち, そのバウンド $O(2^{c/2})$ であること [19] に依拠していると提案者らは述べている [39]. 但し c はキャパシティ長である.

その他, ISAP のブラックボックス安全性に関連する研究として Lefevre と Mennink によるスポンジ構造の衝突, 原像計算, 第二原像計算の困難性の評価 [79] が挙げられる. これは, Bertoni らによって示されたバウンド $O(2^{c/2})$ の Indifferentiability [19] から導出される 3 つの困難性に対する安全性バウンドが, 衝突と第二原像計算についてはタイトであり, 原像計算についてはより良い安全性バウンドを導出できるということを示した結果である.

耐漏洩安全性 ISAP の認証暗号としての耐漏洩安全性は提案者らによって証明されている [36, 43]. 用いられている耐漏洩安全性定義や漏洩モデルの詳細は本報告書では省略するが, 概要として耐漏洩安全性は, 攻撃者が漏洩情報を出力する暗復号オラクルにアクセスできる場合に, 攻撃者が漏洩情報のない Real world (暗復号オラクル) と漏洩情報のない Ideal world (ランダムオラクルとエラーメッセージのみを返すオラクル) を判別できる確率として定義される. この安全性は Romulus のインスタンスの Romulus-T で用いられている耐漏洩安全性定義である CIML2 や CCAmL2 [15] とは異なる定義である. また, 漏洩情報の仮定として Non-adaptive leakage と Bounded leakage model [88] を用いている. 前者は, 攻撃者が全てのクエリを通じて固定された漏洩情報出力関数を用いる設定で, クエリの度に適応的にその関数を選ぶことはできない設定を指す. 後者は, その漏洩情報出力関数から得られる漏洩情報量の上界が定められている設定である. 証明内部は, 提案者らによるいくつかの既存文献の結果がベースとなっている. ISAPRK 及び ISAPENC の安全性は Dobraunig と Mennink による Keyed duplex の耐漏洩安全性の証明 [42], ISAPMAC の安全性は Dobraunig と Mennink による Suffix keyed sponge の耐漏洩安全性の証明 [44] がそれぞれベースとなっている.

表 4.4: Keccak-p[400] に対する解析. 文献 [39] の Table 11 より一部引用.

Type	Target	Rounds	Time	Method	Reference
Distinguisher	Permutation	20 / 20	$2^{396.0}$	Zero-sum	[27]
	Permutation	12 / 20	$2^{396.0}$	Integral	[27]
	Permutation	7 / 20	$2^{84.0}$	Limited-Birthday	[50]
	Permutation	6 / 20	$2^{278.0}$	Differential	[87]

表 4.5: Ketje Sr v1, Ketje Sr v2 に対する解析. 文献 [116] の表 7.5, 及び文献 [39] の Table 9 より一部引用.

Type	Target	Rounds	Time	Method	Reference
Key recovery	Ketje Sr v1	7 / 13	$2^{115.3}$	Cube	[48]
	Ketje Sr v1	7 / 13	$2^{113.6}$	Cube	[99]
	Ketje Sr v1	7 / 13	$2^{91.0}$	Conditional cube	[100]
	Ketje Sr v2	7 / 13	$2^{75.0}$	Conditional cube	[81]
	Ketje Sr v2	7 / 13	$2^{113.9}$	Cube	[48]
	Ketje Sr v2	7 / 13	$2^{99.0}$	Cube	[99]
	Ketje Sr v2	7 / 13	$2^{77.0}$	Conditional cube	[81]

加えて提案者らは Isap が, Nonce misuse での改ざん検知の安全性, 未検証平文が攻撃者に与えられる設定 (Release of unverified plaintext [7]) での安全性, 同一の暗号文から別々の ($\perp$ でない) 復号結果が得られるような鍵のペアが存在しないことを保証する安全性 (Key committing [51]) を持つと主張している [39].

その他, Isap の耐漏洩安全性に関連する研究として Guo らによる Duplex の耐漏洩安全性の証明, 及びそれを用いた認証暗号の提案 [59] が挙げられる. 漏洩モデルは Hard-to-invert [46] と呼ばれるもので Isap の漏洩モデルとは異なるため, Isap 提案者らはこの証明が文献 [36] における Isap の耐漏洩安全性証明の補完となっていると述べている [37].

4.3.3 プリミティブ及び関連方式の安全性

本節では用いるプリミティブへの解析状況についてまとめる. 加えて, Isap と同じプリミティブを用いている方式で, かつ Isap とモード構成が類似する方式に対する解析をまとめる. 但しここで挙げる解析は公開ランダム置換を仮定したモードへの解析ではなく, ラウンド数を削減したプリミティブを用いた方式への解析である.

プリミティブ Ascon-p に対する解析状況の詳細は Ascon の安全性評価レポートを参照されたいが, 現時点における識別攻撃可能段数の最大は 12 段であり, これはゼロサム攻撃によるものである [65]. Isap ではこれより少ないラウンド数の Ascon-p を用いている箇所があるものの, 提案者らは Isap 内で用いる置換に対してランダム置換との識別不可能性は必要ないとしており, 上記の解析は Isap 全体の安全性には影響しないと述べている [39].

次に Keccak-p[400] に対する識別攻撃について表 4.4 にまとめる. 現時点における攻撃可能段数の最大は 20 段で, ゼロサム攻撃によるものである. 但し, 必要な計算量が Isap が主張する安全性レベルを大きく超えていることに注意されたい. Isap ではこれより少ないラウンド数の Keccak-p[400] を用いている箇所があるが, Ascon-p の場合と同じく, 提案者らは Isap 全体の安全性には影響しないと述べている [39].

表 4.6: KECCAK-p[400] を用いた Keccak ハッシュへの解析結果. 文献 [116] の表 5.1, 文献 [39] の Table 10 より一部引用. Keccak$[r, c]$ はスポンジ構造のレートが r ビット, キャパシティが c ビットであることを示す.

Type	Target	Output size	Rounds	Time	Method	Reference
Preimage	Keccak[240,160]	80	3 / 20	$2^{45.0}$	Algebraic	[1,80]
Collision	Keccak[240,160]	160	4 / 20	-	Differential	[1,76]
	Keccak[144,256]	Arbitrary	2 / 20	$2^{101.5}$	Algebraic collision	[25]

ISAP-A-128A 及び ISAP-A-128 の関連方式 ISAP-A-128A 及び ISAP-A-128 に対して提案者らは, ASCON と ASCONHASH の攻撃可能段数から, 内部関数 ISAPENC と ISAPMAC の安全性をそれぞれ測ることができると述べている [39]. ASCON は ASCON-p を内部置換として用いたスポンジ構造の認証暗号であり, ASCONHASH は ASCON-p を内部置換として用いたスポンジ構造のハッシュであって ISAPMAC の AD・暗号文処理部分と類似した構成である.

ASCON と ASCONHASH に対する解析状況の詳細は ASCON の安全性レポートを参照されたいが, 現時点における ASCON の鍵回復攻撃可能段数の最大は 7 段 [96] であり, これは ASCON の初期化フェーズに対する攻撃である. 提案者らは ISAP-A-128A と ISAP-A-128 においては, ナンス処理後から平文処理までに要するラウンド数 $(s_{\mathrm{K}} + s_{\mathrm{E}})$ がそれぞれ 18 段, 24 段と十分に大きいため, 現状の仕様が十分にセキュリティマージンを持つと述べている [39]. また, Nonce respecting 設定下における ASCON のステート回復攻撃可能段数の最大は 3 段 [53] であり, これは ASCON インスタンスの ASCON-128A における平文処理フェーズに対する攻撃である. 提案者らは ISAP-A-128A と ISAP-A-128 においては, 平文処理部分のラウンド数 (s_{E}) がそれぞれ 6 段と 12 段であり, 且つレートが 64 ビットで ASCON-128A の 128 ビットよりも小さいことから, 現状の仕様が十分にセキュリティマージンを持つと述べている [39]. ASCONHASH においては現時点で 2 段 [53,115] までの衝突攻撃が提案されており, ISAP-A-128A と ISAP-A-128 に直接適用できるものの, 仕様上のラウンド数はどちらも 12 段であるため十分なセキュリティマージンがあると提案者らは述べている [39].

ISAP-K-128A 及び ISAP-K-128 の関連方式 ISAP-K-128A 及び ISAP-K-128 に対して提案者らは, Ketje [18] と Keccak [17] ハッシュの攻撃可能段数から, 内部関数 ISAPENC と ISAPMAC の安全性をそれぞれ測ることができると述べている [39]. Ketje は用いる置換が KECCAK-p と同様の構成をしているスポンジ構造の認証暗号であり, 400 ビット入出力の置換を用いたインスタンス KETJE SR v1, KETJE SR v2 が存在する. Keccak ハッシュは ASCONHASH の場合と同様に ISAPMAC の AD・暗号文処理部分に類似しており, KECCAK-p[400] を用いたインスタンスが存在する.

KETJE SR への解析結果を表 4.5 にまとめる. 攻撃可能段数の最大は 7 段であり, これは KETJE SR の初期化フェーズに対する攻撃である. KETJE SR のレートは 32 ビットで ISAPENC の 144 ビットよりも小さいものの, 提案者らは ASCON の場合と同様に ISAP-K-128A と ISAP-K-128 においては, ナンス処理後から平文処理までに要するラウンド数がそれぞれ 16 段, 24 段と十分に大きいため, この解析結果は仕様上の安全性を脅かすものではないと述べている [39].

Keccak ハッシュへの解析結果を表 4.6 にまとめる. 最大 3 段までの原像計算攻撃, 及び 4 段までの衝突攻撃が存在しているが, どちらの結果も ISAP よりも小さいキャパシティのスポンジ構造を用いている. ISAP と同じキャパシティを用いている文献 [25] の攻撃可能段数は 2 段であり, この解析は ISAP-K-128A と ISAP-K-128 に直接適用できるものの, 仕様上のラウンド数は ISAP-K-128A が 16 段, ISAP-K-128 が 20 段であるため十分なセキュリティマージンがあると提案者らは述べている [39].

4.3.4 その他

ISAP や関連する方式に対するサイドチャネル攻撃耐性の解析がいくつか発表されている [10, 75, 104, 111]. 提案者らは ISAP モードの耐漏洩安全性について証明可能安全性を示しているものの, ISAP 全体に対して現実的に実現される具体的な耐漏洩安全性レベルを主張している訳ではないため, 何れも仕様書で主張している安全性と矛盾しない. また, Janson と Struck によって ISAP の暗号化部分に対して量子識別攻撃ができることが示されている [73] が, 提案者らは耐量子安全性については特に主張していないため, この解析も仕様書で主張している安全性とは矛盾しない.

第5章　Romulus

Romulus [55] は NIST LWC ファイナリストの中で唯一, ブロック暗号の拡張である Tweakable ブロック暗号 (TBC) をプリミティブとして用いた方式である．用いる TBC は ISO/IEC 18033-7:2022 [69] において標準化されているものと同様である．モードは複数用意されており, それぞれ達成する安全性の種類が異なる．2022 年 9 月末時点で, 提案者らが仕様書において主張する安全性を損なう解析結果は発表されていない．

5.1　準備

空文字列を ε と定義し, $|\varepsilon| = 0$ とする．$\{0,1\}^0 = \varepsilon$ としたとき, $\{0,1\}^{\leq n} = \cup_{i=0,\ldots,n}\{0,1\}^i$ と定義する．ビット列 $X \in \{0,1\}^*$, 自然数 n について, $|X|_n = \max\{1, \lceil |X|/n \rceil\}$ とする．また, $X[1]\ldots X[x] \xleftarrow{n} X$ は X を n ビットブロックに分割することを指し, $X[1] \| X[2] \| \cdots \| X[x] = X$ かつ $x = |X|_n$ である．もし $X = \varepsilon$ の場合は $X[1] \xleftarrow{n} X$ かつ $X[1] = \varepsilon$ であり, $|\varepsilon|_n = 1$ である．ビット列 $X \in \{0,1\}^n$, 自然数 $x \leq n$ について, $\mathtt{msb}_x(X)$ (resp. $\mathtt{lsb}_x(X)$) を X の上位 (resp. 下位) x ビットのビット列とする．自然数 i に対し, 0^i を 0 を i 個並べたビット列とし, $[\![i]\!]_0 = \{0, 1, \ldots, i-1\}$ とする．l を 8 の倍数とし, $X \in \{0,1\}^{\leq l}$ を長さが 8 の倍数となるビット列 (つまりバイト列) とするとき, パディング関数 pad を以下のように定義する．

$$\mathtt{pad}_l(X) = \begin{cases} X & (|X| = l) \\ X \,\|\, 0^{l-|X|-8} \,\|\, \mathtt{len}_8(X) & (0 \leq |X| < l) \end{cases}$$

ここで, $\mathtt{len}_8(X)$ は 1 バイトの X のバイト長表現である．

5.2　仕様

Romulus は Tweakable ブロック暗号 (TBC) の Skinny-128-384+ をプリミティブとして用いた認証暗号, Romulus-N, Romulus-M, Romulus-T と, Skinny-128-384+ をプリミティブとして用いたハッシュ関数 Romulus-H をまとめた総称である．この 4 方式は全て同じプリミティブを用いるがモード構成が異なる．Primary member は Romulus-N である．

5.2.1　プリミティブ

Skinny-128-384+ はブロックサイズ 128 ビット, Tweakey サイズ 384 ビットの TBC である．Romulus-N, Romulus-M, Romulus-T については鍵長が 128 ビット, Tweak 長が 256 ビットである．Skinny-128-384+ の構成は, CRYPTO 2016 で提案された Skinny [11] のインスタンスの 1 つである Skinny-128-384 のラウンド数を, 56 段から 40 段に削減したものである．なお, Skinny は ISO/IEC 18033-7:2022 において標準化されている [69]．本節では Skinny-128-384+ の暗号化の仕様について述べる．

Skinny-128-384+ の内部ステートは 1 要素が 1 バイトの 4×4 の配列で表される．この配列を IS とおく．$i, j \in \{0, 1, 2, 3\}$ に対し, $IS_{i,j}$ を IS の i 行 j 列における要素とする．また, $i \in \{0, 1, \ldots, 15\}$

に対し, IS を 1 つのベクトルとして見なした場合にその i 番目の要素を IS_i と表記する．つまり $IS_{i,j} = IS_{4i+j}$ である．Tweakey ステートも内部ステートと同様に, 1 要素が 1 バイトの 4×4 の配列 3 つによって表される．この配列を順番に $TK1, TK2, TK3$ とする．$z \in \{1, 2, 3\}$, $i, j \in \{0, 1, 2, 3\}$ に対し, $TKz_{i,j}$ を z 番目の Tweakey ステート配列 TKz の i 行 j 列における要素とする．また, $z \in \{1, 2, 3\}$, $i \in \{0, 1, \ldots, 15\}$ に対し, TKz を 1 つのベクトルとして見なした場合にその i 番目の要素を TKz_i と表記する．入力する 128 ビットの平文 m を $m = m_0 \| m_1 \| \cdots \| m_{15}$ とおく．ここで $i \in \{0, \ldots, 15\}$ に対して m_i は 1 バイトである．入力する 384 ビットの Tweakey tk を $tk = tk_0 \| tk_1 \| \cdots \| tk_{46} \| tk_{47}$ とおく．ここで $i \in \{0, \ldots, 47\}$ に対して tk_i は 8 ビットであり, $TK1_i = tk_i$, $TK2_i = tk_{16+i}$, $TK3_i = tk_{32+i}$ とする．

Skinny-128-384+ のラウンド関数は `SubCells`, `AddConstants`, `AddRoundTweakey`, `ShiftRows`, `MixColumns` の 5 つの手順によって構成される．

`SubCells` では内部ステート配列の各要素に対して, それぞれ以下で定義する Sbox を適用する．$i \in \{0, 1, \ldots, 7\}$, $x_i \in \{0, 1\}$ に対し, (x_7, ..., x_0) を Sbox への入力ビットとする．まず入力ビットに対し以下の変換を施す．

$$(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \to (x_7, x_6, x_5, x_4 \oplus (\overline{x_7 \vee x_6}), x_3, x_2, x_1, x_0 \oplus (\overline{x_3 \vee x_2}))$$

次に以下の通りビット置換を行う．

$$(x_7, x_6, x_5, x_4, x_3, x_2, x_1, x_0) \to (x_2, x_1, x_7, x_6, x_4, x_0, x_3, x_5)$$

上記 2 つの変換を 4 回繰り返す．但し, 最後のビット置換は x_1 と x_2 部分の入れ替えのみである．

`AddConstants` では内部ステート配列とラウンド定数との排他的論理和をとる．ラウンド定数は以下で定義する 6 ビット線形帰還シフトレジスタ (LFSR) を用いて決定される．

$$(\mathrm{rc}_5 \| \mathrm{rc}_4 \| \mathrm{rc}_3 \| \mathrm{rc}_2 \| \mathrm{rc}_1 \| \mathrm{rc}_0) \to (\mathrm{rc}_4 \| \mathrm{rc}_3 \| \mathrm{rc}_2 \| \mathrm{rc}_1 \| \mathrm{rc}_0 \| \mathrm{rc}_5 \oplus \mathrm{rc}_4 \oplus 1)$$

なお $i \in \{0, \ldots, 5\}$ に対して $\mathrm{rc}_i \in \{0, 1\}$ であり, 初期値は $\mathrm{rc}_i = 0$ である．上記 LFSR は各ラウンド開始前に状態を遷移させておき, 以下のように配置する．

$$\begin{bmatrix} c_0 & 0 & 0 & 0 \\ c_1 & 0 & 0 & 0 \\ c_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

ここで, $c_2 = \texttt{0x2}$, $(c_0, c_1) = (0 \| 0 \| 0 \| 0 \| \mathrm{rc}_3 \| \mathrm{rc}_2 \| \mathrm{rc}_1 \| \mathrm{rc}_0, 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| \mathrm{rc}_5 \| \mathrm{rc}_4)$ である．内部ステート配列と上記配列との排他的論理和をとる．

`AddRoundTweakey` では各ラウンドで更新される Tweakey と内部ステートとの排他的論理和をとる．まず, 3 つの Tweakey 配列の 1 行目及び 2 行目と内部ステート配列の 1 行目及び 2 行目の排他的論理和をとる．つまり $i \in \{0, 1\}$, $j \in \{0, 1, 2, 3\}$ に対して $IS_{i,j} = IS_{i,j} \oplus TK1_{i,j} \oplus TK2_{i,j} \oplus TK3_{i,j}$ とする．次に各 Tweakey 配列を更新する．$z \in \{1, 2, 3\}$, $i \in \{0, 1, \ldots, 15\}$ に対し配列 $P_T = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$ を用いて $TKz_i \leftarrow TK_{P_T[i]}$ とする．最後に $TK2$ と $TK3$ の 1 行目及び 2 行目の各要素をそれぞれ LFSR を用いて更新する．適用する LFSR は $TK2$ と $TK3$ でそれぞれ異なり, 以下の通り定義される．

$$TK2 \quad (x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) \to (x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0 \| x_7 \oplus x_5)$$
$$TK3 \quad (x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1 \| x_0) \to (x_0 \oplus x_6 \| x_7 \| x_6 \| x_5 \| x_4 \| x_3 \| x_2 \| x_1)$$

ここで, $i \in \{0, 1, \ldots, 7\}$ に対し $x_i \in \{0, 1\}$ である．$TK1$ に関しては LFSR を適用しない．

`ShiftRows` では, 内部ステート配列の行をそれぞれ右に回転させる. 詳細には, $i \in \{0, 1, \ldots, 15\}$ に対し配列 $P = [0, 1, 2, 3, 7, 4, 5, 6, 10, 11, 8, 9, 13, 14, 15, 12]$ を用いて $IS_i \leftarrow IS_{P[i]}$ とする.

`MixColumns` では以下のバイナリ行列 M と内部ステート行列の各列との積をとる.

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

5.2.2 モード

本節では Romulus に含まれる 4 方式のモード構成について概説した後, Primary member である Romulus-N の仕様について詳細に述べる.

概要 認証暗号の 3 方式はそれぞれ達成する安全性の種類が異なる. それに応じてモードの構成も変わる.

Romulus-N は Nonce-based AEAD であり, Nonce respecting の下安全性が保証される認証暗号である. 構成は CHES 2017 で Chakraborti らによって提案されたブロック暗号ベース認証暗号 COFB モード [29] の TBC ベース版がベースとなっている. なお, COFB についてはその後 Journal of cryptology でも掲載 [31] され, NIST LWC のファイナリスト GIFT-COFB [9] のモード構成にも用いられている.

Romulus-M は Misuse-resistant AE (MRAE) であり, Nonce misuse の下安全性が保証される認証暗号である. 認証暗号全体の構成は SIV [95] がベースとなっており, 入力全体に対してメッセージ認証コード (MAC) を適用した後, 出力されたタグを用いて暗号化を行う構成である. 内部で用いる MAC や暗号化の構成は Romulus-N と同様である.

Romulus-T は Leakage-resilient AE (LRAE) であり, サイドチャネル情報等の暗号内部からの漏洩情報を利用することができる攻撃者に対しても安全性が保証できる認証暗号である. 構成は CHES 2020 で提案された TEDT [15] がベースとなっている. また, 内部で用いるハッシュで Romulus-H を用いている. Romulus-T はサイドチャネル攻撃に対して堅牢となるように設計されているが, ISAP と同様にその耐漏洩安全性を達成するための実装上の要件が存在する. その詳細は文献 [13, 55, 105] を参照されたい.

Romulus-N, Romulus-M, Romulus-T の暗号化関数は入力として k ビット秘密鍵 K, nl ビットナンス N, 任意長 AD A, 任意長平文 M を取り, 出力は $|M|$ ビット暗号文 C, τ ビットタグ T である. 但し仕様上の制約として, 1 回の暗号化 (resp. 復号) につき AD と平文 (resp. 暗号文) は合わせて 2^{59} バイトのサイズまで入力を許すというものがある. つまり, $|A| + |M|$ (resp. $|A| + |C|$) は高々2^{59} バイトである. また Romulus-N, Romulus-M, Romulus-T において, $(k, nl, \tau) = (n, n, n) = (128, 128, 128)$ と定められている. 但し, Romulus-N に関しては必要に応じてタグを規定の 128 ビットから切り詰めて小さくしてもよいが, その場合はタグを切り詰めた分だけ安全性レベルが落ちることに注意されたい. Romulus-M と Romulus-T に関してはタグの全てが復号に必要となるため, タグを切り詰めることはできない.

ハッシュ関数 Romulus-H の構成は Latincrypt 2019 で提案された MDPH [90] と同じ構成であり, その構成は Hirose によって FSE 2006 で提案されたブロック暗号ベースの Double-block ハッシュ関数 [63] と Hirose らによって Asiacrypt 2007 で提案された MDP domain extension 方式 [64] を合わせた構成となっている. 入力として任意長の平文 M を取り, 2τ ビットのハッシュ値 T を出力する. ここで, $\tau = n = 128$ なのでハッシュ値は 256 ビットである.

Algorithm Romulus-N.Enc$_K(N, A, M)$

1. $S \leftarrow 0^n$, $(A[1], \ldots, A[a]) \xleftarrow{n} A$
2. **if** $|A[a]| < n$ **then** $w_A \leftarrow 26$ **else** 24
3. $A[a] \leftarrow \mathtt{pad}_n(A[a])$
4. **for** $i = 1$ **to** $\lfloor a/2 \rfloor$
5. $\quad (S, \eta) \leftarrow \rho(S, A[2i-1])$, $S \leftarrow \widetilde{E}_K^{(A[2i],8,\overline{2i-1})}(S)$
6. **end for**
7. **if** $a \bmod 2 = 0$ **then** $V \leftarrow 0^n$ **else** $A[a]$
8. $(S, \eta) \leftarrow \rho(S, V)$, $S \leftarrow \widetilde{E}_K^{(N,w_A,\overline{a})}(S)$
9. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
10. **if** $|M[m]| < n$ **then** $w_M \leftarrow 21$ **else** 20
11. **for** $i = 1$ **to** $m - 1$
12. $\quad (S, C[i]) \leftarrow \rho(S, M[i])$, $S \leftarrow \widetilde{E}_K^{(N,4,\overline{i})}(S)$
13. **end for**
14. $M'[m] \leftarrow \mathtt{pad}_n(M[m])$, $(S, C'[m]) \leftarrow \rho(S, M'[m])$
15. $C[m] \leftarrow \mathtt{lsb}_{|M[m]|}(C'[m])$, $S \leftarrow \widetilde{E}_K^{(N,w_M,\overline{m})}(S)$
16. $(\eta, T) \leftarrow \rho(S, 0^n)$, $C \leftarrow C[1] \,\|\, \ldots \,\|\, C[m-1] \,\|\, C[m]$
17. **return** (C, T)

Algorithm $\rho(S, M)$

1. $C \leftarrow M \oplus G(S)$, $S' \leftarrow S \oplus M$
2. **return** (S', C)

図 5.1: Romulus-N の暗号化関数 [55]. $\widetilde{E}_K$ は秘密鍵 K を入力した n ビット入出力の TBC である. また, [**if (statement) then** $X \leftarrow x$ **else** x'] は [**if (statement) then** $X \leftarrow x$ **else** $X \leftarrow x'$] の省略である. η はダミー変数でありアルゴリズムには用いられない.

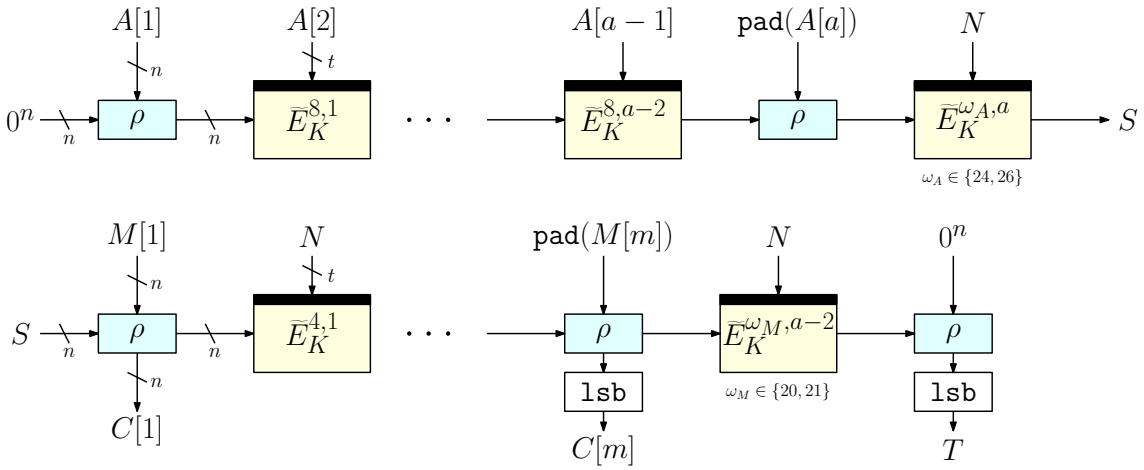


図 5.2: Romulus-N の暗号化. [71] より引用.

詳細　より詳細な定義として, Romulus-N の暗号化アルゴリズムを図. 5.1 に, 図を図 5.2 に示す. Romulus-N の復号関数とその他のインスタンスの詳細なアルゴリズムについては省略する. 図 5.1 中の G は $n \times n$ のバイナリ行列で, 以下のように 8×8 のバイナリ行列 G_s を用いて定義される.

$$G_s = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad G = \begin{pmatrix} G_s & 0 & 0 & \dots & 0 \\ 0 & G_s & 0 & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & 0 & G_s & 0 \\ 0 & \dots & 0 & 0 & G_s \end{pmatrix}.$$

ここで, G の要素として 0 と表記した部分は 8×8 のゼロ行列を表す. 次に Skinny-128-384+ の 384 ビット Tweakey に対する鍵及び Tweak の入力方法について述べる. 秘密鍵 $K \in \{0,1\}^{128}$, ナンス若しくは AD ブロック $T \in \{0,1\}^{128}$, Domain separation 用の数値 $w \in [\![256]\!]_0$, カウンタ $i \in [\![2^{56}-1]\!]_0$ に対して $\widetilde{E}_K^{(T,w,\bar{i})}$ と表記したとき, Skinny-128-384+ の Tweakey には以下の 384 ビットが入力される.

$$\mathsf{lfsr}_{56}(i) \,\|\, (w)_8 \,\|\, 0^{64} \,\|\, T \,\|\, K$$

ここで, $(w)_8$ は w を 8 ビットエンコードした数値とする. $w = 26$ なら $(w)_8 = (00011010)$ である. また, $\mathsf{lfsr}_{56}(i)$ は帰還多項式 $x^{56} + x^7 + x^4 + x^2 + x + 1$ の 56 ビット LFSR で初期値 1 を i 回遷移させたときの 56 ビット値を表す.

5.3　安全性

2022 年 9 月現在, Romulus はいずれの方式についても仕様書で主張される安全性を脅かす攻撃は提案されていない. 本節では, まず 5.3.1 節で提案者らが主張する安全性を概説する. 5.3.2 節で各インスタンスにおけるモードの安全性の根拠及び第三者評価, 5.3.3 節で用いる TBC の解析状況について述べ, 5.3.4 節でその他の安全性解析状況について述べる.

表 5.1: Romulus に含まれる認証暗号 3 方式の安全性. [55] より引用. 単位はビット安全性. n は用いる TBC の入出力長であり, 全ての方式において $n = 128$ である. また, Romulus-T の各安全性数値は Romulus-T を破るために攻撃者が必要なデータ量と時間計算量の和のビット長を示す.

方式	NR-Priv	NR-Auth	NM-Priv	NM-Auth
Romulus-N	n	n	-	-
Romulus-M	n	n	$n/2 \sim n$	$n/2 \sim n$
Romulus-T	$n - \log_2 n$	$n - \log_2 n$	-	$n - \log_2 n$

表 5.2: Romulus-H の各安全性レベル. [55] より引用. 単位はビット安全性. 各数値は Romulus-H の各安全性を破るために攻撃者が必要なデータ量と時間計算量の和のビット長を示す. n は用いる TBC の入出力長であり, $n = 128$ である.

衝突困難性	原像計算困難性	第 2 原像計算困難性
$n - \log_2 n$	$n - \log_2 n$	$n - \log_2 n$

5.3.1 仕様上の安全性

提案者らが主張する Romulus-N, Romulus-M, Romulus-T の安全性の概要を表 5.1 に示す. ここで, NR-Priv, NR-Auth はそれぞれ Nonce respecting の下での秘匿と改ざん検知の安全性を指し, NM-Priv, NM-Auth はそれぞれ Nonce misuse の下での秘匿と改ざん検知の安全性 (Nonce misuse resistance) を指す. また Romulus-M における $n/2 \sim n$ という表記は, 同じナンスが繰り返し用いられる回数に応じて安全性レベルが $n/2$ 以上 n 以下の値となることを指す. 表 5.1 で示される安全性レベルは全て提案者らによって与えられているモードの安全性証明から導出される. Romulus-N, Romulus-M の安全性証明では用いる TBC を TPRP としたときの安全性 [72], Romulus-T の安全性証明では用いる TBC を Ideal cipher としたときの安全性 [57] がそれぞれ示されている. その他にも Romulus-N, Romulus-M, Romulus-T が満たす安全性はいくつか存在するが, 詳細は 5.3.2 節にて述べる.

次に提案者らが主張する Romulus-H の安全性を表 5.2 に示す. この安全性は Naito によって示されている MDPH の安全性証明 [90], 及び提案者らによるその改良 [58] により導出される. これらの安全性証明では用いる (Tweakable) ブロック暗号は Ideal cipher として証明が示される.

5.3.2 モードの安全性

前節で述べた通り, Romulus は全てのインスタンスにおいてモードの安全性証明が示されている. 本節では, Romulus 各インスタンスのモードに対して証明されている安全性やその根拠についてまとめた後, 第三者評価について述べる.

Romulus-N の Nonce respecting 設定下における証明 [72] は iCOFB [30] と PFB [91] の安全性証明がベースとなっている. また, 表 5.1 で示した通り Romulus-N は NM-Priv, NM-Auth の安全性 (Nonce-misuse resistant) が担保出来ないが, これらの安全性定義を弱めた Nonce-misuse resilience [8] という安全性であれば安全性が示せることが示されている [67]. この安全性は, ナンスを誤用した場合であっても, 誤用していないナンスが含まれるクエリに関してはその安全性を担保できるというもので, Romulus-N については Nonce-misuse resilience 設定下で, 秘匿については n ビット, 改ざん検知についてはナンス誤用の回数に応じて $n/2$ ビット以上 n ビット以下の安全性を達成できることが示されている.

Romulus-M は Nonce respecting と Nonce misuse の安全性証明がそれぞれ与えられている [72]. Nonce misuse に関する安全性証明では, 同じナンスが繰り返し使われるほど安全性が n ビットから

$n/2$ ビットへ減少する (Graceful degradation [93]) ことが示されている．また表 5.1 記載の安全性以外のものについては, 未検証平文が攻撃者に与えられる設定 (Release of unverified plaintext [7], 以下 RUP) における安全性について, 提案者らによって証明が与えられている [70]．まず RUP 設定下における改ざん検知の安全性, 通称 INT-RUP 安全性に関しては, Nonce respecting と Nonce misuse 両方の設定において, RUP でない場合とほぼ同様の証明ができ, 同じレベルの安全性が担保できることが示されている．次に RUP 設定下における秘匿の安全性, 通称 plaintext awareness に関しては, PA2 と呼ばれる強い方の安全性は満たさないことが文献 [55] にて述べられている．しかし PA1 と呼ばれる弱い方の安全性については, 入力平文長が n ビットの倍数の場合に $n/2$ ビットの安全性レベルが担保できることが示されている．

Romulus-T の安全性証明は文献 [57] で与えられる．その証明では, Romulus-H が 2 つの耐漏洩安全性 CIML2, CCAmL2 [15] を持ち, その安全性レベルがそれぞれ $n - \log n$ ビット, $n/2$ ビットであることが示されている．詳細な安全性定義及び仮定する漏洩モデルについては [57] を参照されたいが, 概要として 2 つの耐漏洩安全性は以下のような意味を持つ．CIML2 は Ciphertext integrity with misuse-resistance and leakage の略で, Nonce misuse (resistance) 設定かつ暗復号オラクルから漏洩情報を得られる攻撃者を仮定した下での改ざん検知の安全性を指す．CCAmL2 は Chosen-ciphertext attack with misuse-resilience and leakage の略で, Nonce misuse (resilience) 設定かつ暗復号オラクルから漏洩情報を得られる攻撃者を仮定した下での秘匿の安全性を指す．加えて同文献では, ブラックボックス設定下においても Romulus-T が Nonce-misuse resilience の秘匿の安全性を持つことが示されており, その安全性レベルが $n - \log n$ ビットであることが証明されている．

Romulus-H の安全性は MDPH の安全性証明 [58, 90] より導出される．その証明では, 用いる TBC を Ideal cipher としたとき, MDPH がランダムオラクルとの Indifferentiability を持ち, そのバウンドが $O(2^n/n)$ であることが示されている．表 5.2 における安全性レベルはここから導出される．

次に第三者評価について述べる．Romulus の各モードへの第三者評価としては以下が挙げられるが, 仕様上の安全性を脅かすものではない．

- Lee による Romulus-N と Romulus-M の安全性証明に対する第三者評価 [78]．提案者らと独立して Romulus-N の Nonce respecting 下での安全性証明, Romulus-M の Nonce respecting, 及び Nonce misuse の下での安全性証明を行い, それらと提案者らが主張する安全性レベルに齟齬がないことを示している．

- 土生らによる Romulus-N と Romulus-M に対する識別攻撃及び偽造攻撃 [118]．Romulus-N に対する Nonce respecting の下での識別攻撃と偽造攻撃, 及び Romulus-M に対する Nonce misuse の下での識別攻撃と偽造攻撃を提案した．これらの攻撃に必要な計算量が, 安全性証明により示される安全性バウンドが要求する攻撃成功に必要な計算量とそれぞれ同等であるため, それらのバウンドがタイトであることを示す結果である．

- Habu らによる Romulus-M に対するマッチング攻撃 [60]．Romulus-M に対する Nonce misuse の下での識別攻撃と偽造攻撃を提案した．これらの攻撃に必要な計算量が, 安全性証明により示される安全性バウンドが要求する攻撃成功に必要な計算量と同等であるため, それらのバウンドがタイトであることを示す結果である．また同文献において提案した偽造攻撃により, Romulus-M の INT-RUP 安全性バウンドがタイトであることも示されている．

5.3.3 プリミティブの安全性

2022 年 9 月現在, Skinny-128-384+ の安全性を脅かす攻撃は発表されていない．Skinny-128-384+ は Skinny-128-384 のラウンド数を 56 段から 40 段に削減したものであるが, Skinny-128-384 に対する

表 5.3: Skinny-128-384 に対する Related-key 設定下での鍵回復攻撃.

Rounds	Time	Data	Method	Ref
30 / 56	$2^{361.7}$	$2^{125.3}$	Rectangle	[61]
30 / 56	$2^{341.1}$	$2^{122.0}$	Rectangle	[94]
32 / 56	$2^{355.0}$	$2^{123.5}$	Rectangle	[49]
32 / 56	$2^{344.8}$	$2^{123.5}$	Rectangle	[101]

表 5.4: Skinny-128-384 に対する Single-key 設定下での鍵回復攻撃. 但し, 文献 [62] は 360 ビット Single-key で Chosen tweak 設定.

Rounds	Time	Data	Method	Ref
23 / 56	$2^{376.0}$	$2^{104.0}$	MITM	[47]
22 / 56	$2^{376.0}$	$2^{32.0}$	MITM	[66]
25 / 56	$2^{372.5}$	$2^{122.3}$	Diff-MITM	[28]
26 / 56	$2^{344.0}$	$2^{121.0}$	Integral	[62]

現時点での最良解析として, 32 段の Skinny-128-384 に対する Related-key 設定下での Rectangle 攻撃 [49, 101] が挙げられる. また現時点における 128 ビット鍵の Skinny-128-384+ に対する攻撃可能段数として, 提案者らは Related-key 設定下の識別攻撃については 25 段, Single-key 設定下の鍵回復攻撃については 23 段未満であると述べている [56].

本節では主に Skinny-128-384 に対して 2022 年に発表された解析, 及び Related-key 設定と Single-key 設定においてそれぞれ現時点での最良の解析をまとめる.

Related-key 設定 Related-key 設定における Skinny-128-384 への鍵回復攻撃を表 5.3 にまとめる. 但し, これらの解析は Skinny-128-384 の 384 ビット鍵に対する攻撃であり, 128 ビット鍵を持つ Skinny-128-384+ に対するものではないことに注意されたい. より小さい鍵長を持つ Skinny への解析例として, Skinny-128-384 の 1 つの Tweakey 配列 TK を固定した場合の解析で以下のものが挙げられるが, これらも 256 ビット鍵への攻撃であるため 128 ビット鍵を持つ Skinny-128-384+ に対する解析とはならない [56].

- Hadipour らによる Rectangle 攻撃 [61]. 24 段に削減した Skinny-128-256 を時間計算量 $2^{209.9}$, データ量 $2^{125.2}$ で攻撃できることを示した.
- Qin らによる Rectangle 攻撃 [94]. 25 段に削減した Skinny-128-256 を時間計算量 $2^{226.4}$, データ量 $2^{124.5}$ で攻撃できることを示した.
- Dong らによる Rectangle 攻撃 [49]. 26 段に削減した Skinny-128-256 を時間計算量 $2^{254.4}$, データ量 $2^{126.5}$ で攻撃できることを示した.

また, 識別攻撃は以下の通りである.

- Hadipour らによる Boomerang 攻撃 [61]. 25 段に削減した Skinny-128-384 を確率 $2^{-116.6}$ で識別できることを示した. また, 1 つの TK を固定した設定の下では 21 段に削減した Skinny-128-256 に対して確率 $2^{-114.1}$ で識別できることを示した.
- Delaune らによる Boomerang 攻撃 [35]. 24 段に削減した Skinny-128-384 を確率 $2^{-86.1}$ で識別できることを示した. また, 1 つの TK を固定した設定の下では 20 段に削減した Skinny-128-256 に対して確率 $2^{-85.8}$ で識別できることを示した.

Single-key 設定 Single-key 設定における鍵回復攻撃を表 5.4 にまとめる. 但し Related-key 設定の場合と同様に, これらの攻撃に必要な計算量は 2^{128} を大きく超えるため, Skinny-128-384+ の 128 ビット安全性に対する攻撃にはならないことに注意されたい. 384 ビット未満の安全性レベルについて, 提案者らは Skinny-128-256 の 256 ビット安全性に対して計算量 2^{216} で 22 段まで攻撃可能であり, Skinny-64-192 の 128 ビット安全性に対して計算量 $2^{116.5}$ で 17 段まで攻撃可能であると述べている [56].

その他　上に挙げた解析以外の最近の結果としては, Kuijsters らによる線形特性の評価 [77] が挙げられる. 2 ラウンドの Subtweakey[1]の部分集合に対して, 線形特性が 1 のものを発見したという結果であるが, 提案者らはこの解析結果は特に驚くべきものではなく, Skinny の安全性には影響しないと述べている [56]. また, 量子計算機を用いた解析として David らによる 21 段 Skinny-128-256 への量子不能差分攻撃 [34] や, Bijwe らによる Grover アルゴリズムの量子コスト評価 [23] が挙げられるが, 何れも安全性を脅かすものではない.

5.3.4　その他

モード単体, プリミティブ単体の解析以外のものとしては以下が挙げられるが, 仕様上の安全性を脅かすものではない.

- Dong らによるラウンド数を削減し Skinny-128-384+ を用いた Romulus-H に対する原像攻撃及び Romulus-H の内部圧縮関数に対する衝突攻撃 [47]. Skinny-128-384+ フルラウンド 40 段のうち 23 段に削減した場合に, Romulus-H に対して時間計算量 2^{248} で原像攻撃が可能であることを示している. この計算量は提案者らが主張する Romulus-H の安全性レベルを大きく超えているため, 主張される安全性と矛盾しない. また, 23 段に削減した Romulus-H の内部圧縮関数に対して, Free-start 設定下における衝突攻撃が時間計算量 2^{124}, 空間計算量 2^{124} で可能であることを示している.
- Nageler らによるラウンド数を削減した Skinny-128-384+ を用いた Romulus-H での衝突発見 [89]. Skinny-128-384+ フルラウンド 40 段のうち 10 段に削減した場合に衝突を, 14 段に削減した場合に Semi-free-start 設定での衝突を現実的な時間で発見している.

また, Romulus-N のサイドチャネル攻撃に対する安全性解析として, Vialar によるサイドチャネル攻撃を用いた Romulus-N の鍵回復 [107], Chakraborty らによる投機的実行を利用した Romulus-N の鍵回復 [32] が挙げられる. また, 文献 [111] 内に Romulus-N のサイドチャネル情報を評価したレポートが存在する. 但し, 提案者らは Romulus-N の非ブラックボックス安全性については特に主張していないため, これらの解析も仕様書で主張される安全性とは矛盾しない.

[1] `AddRoundTweakey` において内部ステートと排他的論理和を取る Tweakey 部分のことを指す.

参考文献

[1] The Keccak crunchy crypto collision and pre-image contest. https://keccak.team/crunchy_contest.html.

[2] Lightweight Cryptography. http://competitions.cr.yp.to/caesar.html.

[3] Elephant, 2022. https://www.esat.kuleuven.be/cosic/elephant/.

[4] ISAP, 2022. https://isap.iaik.tugraz.at/index.html.

[5] FIPS 202. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015.

[6] Gorjan Alagic, Chen Bai, Jonathan Katz, Christian Majenz, and Patrick Struck. Post-quantum security of the (tweakable) fx construction, and applications. Cryptology ePrint Archive, Paper 2022/1097, 2022. https://eprint.iacr.org/2022/1097.

[7] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In *ASIACRYPT (1)*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[8] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2017.

[9] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB v1.1. Submission to the NIST Lightweight Cryptography project, 2021.

[10] Lejla Batina, Ileana Buhan, Lukasz Chmielewski, Ellen Gunnarsdóttir, Vahid Jahandideh, Tom Stock, and Léo Weissbart. Side-Channel Evaluation Report on Implementations of Several NIST LWC Finalists. 2022. https://cryptography.gmu.edu/athena/LWC/Reports/Radboud/Radboud_Report_SW_3_candidates.pdf.

[11] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 123–153. Springer, Heidelberg, August 2016.

[12] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor,

ASIACRYPT 2000, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000.

[13] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 369–400. Springer, Heidelberg, August 2020.

[14] Daniel J. Bernstein. How to stretch random functions: The security of protected counter sums. *Journal of Cryptology*, 12(3):185–192, June 1999.

[15] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT: a leakage-resistant AEAD mode. *IACR TCHES*, 2020(1):256–320, 2019. `https://tches.iacr.org/index.php/TCHES/article/view/8400`.

[16] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. Ecrypt Hash Workshop 2007, 2007. `http://sponge.noekeon.org/SpongeFunctions.pdf`.

[17] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, January 2011.

[18] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. CAESAR submisiion: Ketje v2, 2016.

[19] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, Heidelberg, April 2008.

[20] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, Heidelberg, August 2012.

[21] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant v2.0. Submission to the NIST Lightweight Cryptography project, 2021.

[22] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Multi-user security of the elephant v2 authenticated encryption mode. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography*, pages 155–178, Cham, 2022. Springer International Publishing.

[23] Subodh Bijwe, Amit Kumar Chauhan, and Somitra Kumar Sanadhya. Implementing grover oracle for lightweight block ciphers under depth constraints. In *ACISP*, volume 13494 of *Lecture Notes in Computer Science*, pages 85–105. Springer, 2022.

[24] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES 2011*, volume 6917 of *LNCS*, pages 312–325. Springer, Heidelberg, September / October 2011.

[25] Rachelle Heim Boissier, Camille Noûs, and Yann Rotella. Algebraic collision attacks on Keccak. *IACR Trans. Symm. Cryptol.*, 2021(1):239–268, 2021.

[26] Xavier Bonnetain and Samuel Jaques. Quantum period finding against symmetric primitives in practice. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):1–27, 2022.

[27] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In Antoine Joux, editor, *FSE 2011*, volume 6733 of *LNCS*, pages 252–269. Springer, Heidelberg, February 2011.

[28] Christina Boura, Nicolas David, Patrick Derbez, Gregor Leander, and María Naya-Plasencia. Differential meet-in-the-middle cryptanalysis. Cryptology ePrint Archive, Paper 2022/1640, 2022. `https://eprint.iacr.org/2022/1640`.

[29] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *CHES 2017*, volume 10529 of *Lecture Notes in Computer Science*, pages 277–298. Springer, 2017.

[30] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? (full version of [29]). *IACR Cryptology ePrint Archive*, 2017:649, 2017.

[31] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? *Journal of Cryptology*, 33(3):703–741, July 2020.

[32] Anirban Chakraborty, Nikhilesh Singh, Sarani Bhattacharya, Chester Rebeiro, and Debdeep Mukhopadhyay. Timed speculative attacks exploiting store-to-load forwarding bypassing cache-based countermeasures. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, DAC '22, page 553–558, New York, NY, USA, 2022. Association for Computing Machinery.

[33] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part II*, volume 10625 of *LNCS*, pages 606–637. Springer, Heidelberg, December 2017.

[34] Nicolas David, María Naya-Plasencia, and André Schrottenloher. Quantum impossible differential attacks: Applications to aes and skinny. Cryptology ePrint Archive, Paper 2022/754, 2022. `https://eprint.iacr.org/2022/754`.

[35] Stéphanie Delaune, Patrick Derbez, and Mathieu Vavrille. Catching the fastest boomerangs application to SKINNY. *IACR Trans. Symmetric Cryptol.*, 2020(4):104–129, 2020.

[36] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symm. Cryptol.*, 2020(S1):390–416, 2020.

[37] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. NIST update: ISAP v2.0, 2020. `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/ISAP_update_isap.pdf`.

[38] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. Submission to the NIST Lightweight Cryptography project, 2021.

[39] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. NIST update: ISAP v2.0, 2022. `https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/isap-update.pdf`.

[40] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP – towards side-channel secure authenticated encryption. *IACR Trans. Symm. Cryptol.*, 2017(1):80–105, 2017.

[41] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to the NIST Lightweight Cryptography project, 2021.

[42] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 225–255. Springer, Heidelberg, December 2019.

[43] Christoph Dobraunig and Bart Mennink. Leakage resilience of the ISAP mode: a vulgarized summary. NIST Lightweight Cryptography Workshop 2019, 2019.

[44] Christoph Dobraunig and Bart Mennink. Security of the suffix keyed sponge. *IACR Trans. Symm. Cryptol.*, 2019(4):223–248, 2019.

[45] Christoph Dobraunig and Bart Mennink. Tightness of the suffix keyed sponge bound. *IACR Trans. Symm. Cryptol.*, 2020(4):195–212, 2020.

[46] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 621–630. ACM Press, May / June 2009.

[47] Xiaoyang Dong, Jialiang Hua, Siwei Sun, Zheng Li, Xiaoyun Wang, and Lei Hu. Meet-in-the-middle attacks revisited: Key-recovery, collision, and preimage attacks. In *CRYPTO (3)*, volume 12827 of *Lecture Notes in Computer Science*, pages 278–308. Springer, 2021.

[48] Xiaoyang Dong, Zheng Li, Xiaoyun Wang, and Ling Qin. Cube-like attack on round-reduced initialization of Ketje Sr. *IACR Trans. Symm. Cryptol.*, 2017(1):259–280, 2017.

[49] Xiaoyang Dong, Lingyue Qin, Siwei Sun, and Xiaoyun Wang. Key guessing strategies for linear key-schedule algorithms in rectangle attacks. In *EUROCRYPT (3)*, volume 13277 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2022.

[50] Alexandre Duc, Jian Guo, Thomas Peyrin, and Lei Wei. Unaligned rebound attack: Application to Keccak. In Anne Canteaut, editor, *FSE 2012*, volume 7549 of *LNCS*, pages 402–421. Springer, Heidelberg, March 2012.

[51] Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symm. Cryptol.*, 2017(1):449–473, 2017.

[52] Thomas Fuhr, María Naya-Plasencia, and Yann Rotella. State-recovery attacks on modified Ketje Jr. *IACR Trans. Symmetric Cryptol.*, 2018(1):29–56, 2018.

[53] David Gerault, Thomas Peyrin, and Quan Quan Tan. Exploring differential-based distinguishers and forgeries for ASCON. *IACR Trans. Symm. Cryptol.*, 2021(3):102–136, 2021.

[54] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 263–293. Springer, Heidelberg, May 2016.

[55] Chun Guo, Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Romulus v1.3. Submission to the NIST Lightweight Cryptography project, 2021.

[56] Chun Guo, Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Final-round updates on Romulus, 2022. `https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/romulus-update.pdf`.

[57] Chun Guo, Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Security Proof for Romulus-T. `https://romulusae.github.io/romulus/docs/Romulus_T_proof.pdf`, 2022.

[58] Chun Guo, Tetsu Iwata, and Kazuhiko Minematsu. New indifferentiability security proof of MDPH hash function. *IET Inf. Secur.*, 16(4):262–281, 2022.

[59] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant AE from the duplex sponge. *IACR Trans. Symm. Cryptol.*, 2020(1):6–42, 2020.

[60] Makoto Habu, Kazuhiko Minematsu, and Tetsu Iwata. Matching attacks on Romulus-M. *IET Inf. Secur.*, 16(6):459–469, 2022.

[61] Hosein Hadipour, Nasour Bagheri, and Ling Song. Improved rectangle attacks on SKINNY and CRAFT. *IACR Trans. Symmetric Cryptol.*, 2021(2):140–198, 2021.

[62] Hosein Hadipour, Sadegh Sadeghi, and Maria Eichlseder. Finding the impossible: Automated search for full impossible differential, zero-correlation, and integral attacks (preliminary version). Cryptology ePrint Archive, Paper 2022/1147, 2022. `https://eprint.iacr.org/2022/1147`.

[63] Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In *FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[64] 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.

[65] Kai Hu and Thomas Peyrin. Revisiting higher-order differential(-linear) attacks from an algebraic perspective – applications to ascon, grain v1, xoodoo, and chacha. Cryptology ePrint Archive, Paper 2022/1335, 2022. `https://eprint.iacr.org/2022/1335`.

[66] Jialiang Hua, Tai Liu, Yulong Cui, Lingyue Qin, Xiaoyang Dong, and Huiyong Cui. Low-Data Cryptanalysis On SKINNY Block Cipher. *The Computer Journal*, 02 2022. bxab208.

[67] Akiko Inoue, Chun Guo, and Kazuhiko Minematsu. Nonce-misuse resilience of romulus-n and gift-cofb. Cryptology ePrint Archive, Paper 2022/1012, 2022. `https://eprint.iacr.org/2022/1012`.

[68] ISO/IEC. 29192-5:2016 – Information technology – Security techniques – Lightweight cryptography – Part 5: Hash-functions, 2016.

[69] ISO/IEC. 18033-7:2022 – Information security – Encryption algorithms – Part 7: Tweakable block ciphers, 2022.

[70] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. New Results on Romulus. NIST Lightweight Cryptography Workshop 2020, 2006. Available at `https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/new-results-romulus-lwc2020.pdf`.

[71] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Updates on Romulus, Remus and TGIF. NIST Lightweight Cryptography Workshop 2019, 2019.

[72] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[73] Christian Janson and Patrick Struck. Sponge-based authenticated encryption: Security against quantum attackers. In *PQCrypto*, volume 13512 of *Lecture Notes in Computer Science*, pages 230–259. Springer, 2022.

[74] Priyanka Joshi and Bodhisatwa Mazumdar. Single event transient fault analysis of ELEPHANT cipher. *CoRR*, abs/2106.09536, 2021.

[75] Matthias J. Kannwischer, Peter Pessl, and Robert Primas. Single-trace attacks on Keccak. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):243–268, 2020.

[76] Stefan Kölbl, Florian Mendel, Tomislav Nad, and Martin Schläffer. Differential cryptanalysis of Keccak variants. In Martijn Stam, editor, *14th IMA International Conference on Cryptography and Coding*, volume 8308 of *LNCS*, pages 141–157. Springer, Heidelberg, December 2013.

[77] Daniël Kuijsters, Denise Verbakel, and Joan Daemen. Weak subtweakeys in SKINNY. Cryptology ePrint Archive, Paper 2022/1042, 2022. `https://eprint.iacr.org/2022/1042`, To appear in INDOCRYPT 2022.

[78] Jooyoung Lee. Security evaluation of romulus. `https://romulusae.github.io/romulus/docs/Security_evaluation_Romulus_Jooyoung_Lee.pdf`.

[79] Charlotte Lefevre and Bart Mennink. Tight preimage resistance of the sponge construction. In *CRYPTO (4)*, volume 13510 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.

[80] Ting Li, Yao Sun, Maodong Liao, and Dingkang Wang. Preimage attacks on the round-reduced Keccak with cross-linear structures. *IACR Trans. Symm. Cryptol.*, 2017(4):39–57, 2017.

[81] Zheng Li, Xiaoyang Dong, Wenquan Bi, Keting Jia, Xiaoyun Wang, and Willi Meier. New conditional cube attack on Keccak keyed modes. *IACR Trans. Symm. Cryptol.*, 2019(2):94–124, 2019.

[82] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002.

[83] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, July 2011.

[84] Atul Luykx, Bart Preneel, Elmar Tischhauser, and Kan Yasuda. A MAC mode for lightweight block ciphers. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 43–59. Springer, Heidelberg, March 2016.

[85] Hasindu Madushan, Iftekhar Salam, and Janaka Alawatugoda. A review of the NIST lightweight cryptography finalists and their fault analyses. *Electronics*, 11(24), 2022.

[86] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 279–296. Springer, Heidelberg, May 2010.

[87] Silvia Mella, Joan Daemen, and Gilles Van Assche. New techniques for trail bounds and application to differential trails in Keccak. *IACR Trans. Symm. Cryptol.*, 2017(1):329–357, 2017.

[88] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer, Heidelberg, February 2004.

[89] Marcel Nageler, Felix Pallua, and Maria Eichlseder. Finding collisions for round-reduced romulus-h. Cryptology ePrint Archive, Paper 2022/1630, 2022. `https://eprint.iacr.org/2022/1630`.

[90] Yusuke Naito. Optimally indifferentiable double-block-length hashing without post-processing and with support for longer key than single block. In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 65–85. Springer, 2019.

[91] Yusuke Naito and Takeshi Sugawara. Lightweight authenticated encryption mode of operation for tweakable block ciphers. *IACR TCHES*, 2020(1):66–94, 2019. `https://tches.iacr.org/index.php/TCHES/article/view/8393`.

[92] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, Heidelberg, May 2014.

[93] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 33–63. Springer, Heidelberg, August 2016.

[94] Lingyue Qin, Xiaoyang Dong, Xiaoyun Wang, Keting Jia, and Yunwen Liu. Automated search oriented to key recovery on ciphers with linear key schedule applications to boomerangs in SKINNY and forkskinny. *IACR Trans. Symmetric Cryptol.*, 2021(2):249–291, 2021.

[95] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006.

[96] Raghvendra Rohit, Kai Hu, Sumanta Sarkar, and Siwei Sun. Misuse-free key-recovery and distinguishing attacks on 7-round Ascon. *IACR Trans. Symm. Cryptol.*, 2021(1):130–155, 2021.

[97] André Schrottenloher and Marc Stevens. Simplified mitm modeling for permutations: New (quantum) attacks. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, pages 717–747, Cham, 2022. Springer Nature Switzerland.

[98] Tairong Shi, Wenling Wu, Bin Hu, Jie Guan, and Sengpeng Wang. Breaking LWC candidates: sESTATE and Elephant in quantum setting. *Des. Codes Cryptogr.*, 89(7):1405–1432, 2021.

[99] Ling Song and Jian Guo. Cube-attack-like cryptanalysis of round-reduced KECCAK using MILP. *IACR Trans. Symm. Cryptol.*, 2018(3):182–214, 2018.

[100] Ling Song, Jian Guo, Danping Shi, and San Ling. New MILP modeling: Improved conditional cube attacks on Keccak-based constructions. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 65–95. Springer, Heidelberg, December 2018.

[101] Ling Song, Nana Zhang, Qianqian Yang, Danping Shi, Jiahao Zhao, Lei Hu, and Jian Weng. Optimizing rectangle attacks: A unified and generic framework for key recovery. *IACR Cryptol. ePrint Arch.*, page 723, 2022.

[102] Ling Sun, Wei Wang, and Meiqin Wang. Milp-aided bit-based division property for primitives with non-bit-permutation linear layers. *IET Inf. Secur.*, 14(1):12–20, 2020.

[103] Shu Takemoto, Yoshiya Ikezaki, Yusuke Nozaki, and Masaya Yoshikawa. Hardware trojan for lightweight cryptoraphy elephant. In *2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, pages 944–945, 2021.

[104] Balazs Udvarhelyi, Olivier Bronchain, and François-Xavier Standaert. Security analysis of deterministic re-keying with masking and shuffling: Application to ISAP. In *COSADE*, volume 12910 of *Lecture Notes in Computer Science*, pages 168–183. Springer, 2021.

[105] Corentin Verhamme, Gaëtan Cassiers, and François-Xavier Standaert. Analyzing the leakage resistance of the nist's lightweight crypto competition's finalists. Cryptology ePrint Archive, Paper 2022/1628, 2022. `https://eprint.iacr.org/2022/1628`, , To appear in CARDIS 2022.

[106] Louis Vialar. Fast side-channel key-recovery attack against elephant dumbo. Cryptology ePrint Archive, Paper 2022/446, 2022. https://eprint.iacr.org/2022/446.

[107] Louis Vialar. *Side Channel Analysis of NIST Lightweight Cryptography Candidates*. Master's thesis, Swiss Federal Institute of Technology in Lausanne, 2022.

[108] Hailun Yan, Xuejia Lai, Lei Wang, Yu Yu, and Yiran Xing. New zero-sum distinguishers on full 24-round Keccak-f using the division property. *IET Inf. Secur.*, 13(5):469–478, 2019.

[109] Guo-Shuang Zhang, Yin Li, and Xiao Chen. Improved state-recovery attacks on modified KETJE JR. 2021.

[110] Guoyan Zhang and Meicheng Liu. A distinguisher on PRESENT-like permutations with application to SPONGENT. Cryptology ePrint Archive, Report 2016/236, 2016. https://eprint.iacr.org/2016/236.

[111] Xiaolin Zhang, Tengfei Wang, and Pei Cao. Side-Channel Evaluation on Protected Implementations of Several NIST LWC Finalists. 2022. https://cryptography.gmu.edu/athena/LWC/Reports/SJTU/SJTU_Report_HW_4_candidates_RUB.pdf.

[112] Zishen Zhao, Shiyao Chen, Meiqin Wang, and Wei Wang. Improved cube-attack-like cryptanalysis of reduced-round Ketje-Jr and Keccak-MAC. *Inf. Process. Lett.*, 171:106124, 2021.

[113] Haibo Zhou, Zheng Li, Xiaoyang Dong, Keting Jia, and Willi Meier. Practical key-recovery attacks on round-reduced Ketje Jr, Xoodoo-AE and xoodyak. *Comput. J.*, 63(8):1231–1246, 2020.

[114] Haibo Zhou, Rui Zong, Xiaoyang Dong, Keting Jia, and Willi Meier. Interpolation attacks on round-reduced elephant, kravatte and xoofff. Cryptology ePrint Archive, Report 2020/781, 2020. https://eprint.iacr.org/2020/781.

[115] Rui Zong, Xiaoyang Dong, and Xiaoyun Wang. Collision attacks on round-reduced gimli-hash/ascon-xof/ascon-hash. Cryptology ePrint Archive, Paper 2019/1115, 2019. https://eprint.iacr.org/2019/1115.

[116] 伊藤竜馬. 「CRYPTREC 暗号技術ガイドライン（軽量暗号）」掲載の暗号方式に関する安全性評価の動向調査（文書番号: CRYPTREC EX-3101-2021）, 2022. https://www.cryptrec.go.jp/exreport/cryptrec-ex-3101-2021.pdf.

[117] 土生亮 and 岩田哲. Elephant に対する鍵回復, 識別及び偽造攻撃. SCIS2022, 1F2-5, 2022.

[118] 土生亮, 峯松一彦, and 岩田哲. Romulus-N 及び Romulus-M に対する識別攻撃及び偽造攻撃. **信学技報**, 121(22, ISEC2021-6):25–31, 2022.