CRYPTREC EX-3202-2022

軽量暗号の安全性に関する調査及び評価
(GIFT-COFB, Xoodyak)

三菱電機株式会社
内藤　祐介

2022 年 12 月

エグゼクティブサマリー

NIST が主催の Lightweight Cryptography Standardization Process[19] の最終候補である GIFT-COFB [2, 3, 1] と Xoodyak [11, 10] の安全性に関する文献の調査結果をまとめる．

GIFT-COFB の安全性に関する文献調査　GIFT-COFB は，ブロック暗号ベースの認証暗号である．設計者が主張する安全性は，シングルユーザで，秘匿性について，オンライン計算で 64 bit，オフライン計算で 112 bit 以上，偽造不可能性について，オンライン計算で 58 bit，オフライン計算で 112 bit 以上である．GIFT-COFB で用いられているブロック暗号 GIFT-128 と，GIFT-COFB の安全性に関する文献調査の結果は以下の通りある．

GIFT-128 の安全性　GIFT-128 は，鍵サイズとブロックサイズが 128 bit，40 ラウンドのブロック暗号である [5]．これまでの最適な攻撃は，差分攻撃法で，ラウンド数を 27 ラウンドに削減した GIFT-128 を破ることができる [25]．GIFT-128 のラウンド数は 40 ラウンド，攻撃可能なラウンド数は 27 ラウンドと十分にマージンがあるため，GIFT-128 はオフライン計算に関して設計者が主張する 112 bit 以上の安全性を持つと考えられる．

GIFT-COFB の安全性　GIFT-COFB は，利用モードとして文献 [8] の COFB を用いている．COFB の安全性では，ブロック暗号を Pseudo-Random Permutation (PRP) と仮定している．上記の通り，GIFT-128 の安全性は担保されているため，GIFT-COFB の安全性は COFB の安全性に依存する．文献 [8, 4] で，ブロックサイズを n bit，ブロック暗号を PRP 安全と仮定すると，シングルユーザに対して，秘匿性のオンライン計算に対して $n/2$ bit，偽造不可能性のオンライン計算に対して $n/2 - \log_2 n$ bit となることが証明されている．GIFT-COFB は $n = 128$ であり，上記の通り，GIFT-128 はオフライン計算に関して 112 bit 以上の安全性を持つことから，設計者が主張する安全性を持つと考えられる．

Xoodyak の安全性に関する調査　Xoodyak は，置換ベースのアルゴリズムであり，ハッシュ関数と認証暗号の 2 つのアルゴリズムを備える．設計者が主張する安全性は，Collision Resistance，Second Preimage Resistance，Preimage Resistance に関して 128 bit，m-Target Preimage Resistance に関して $\min\{256 - \log_2 m, 128\}$ bit，認証暗号について，シングルユーザで，秘匿性と偽造不可能性について，オンライン計算で 160bit，オフライン計算で 128bit である．Xoodyak で用いられる置換 Xoodoo[12] と Xoodyak の安全性に関する文献調査の結果は以下の通りある．

Xoodoo[12] の安全性　Xoodoo[12] は 384 bit，12 ラウンドの置換である．Xoodoo[12] 単体では，文献 [16] で，オフライン計算量が 2^{33} の Zero-sum Distinguisher が提案されており，Xoodyak の設計者が主張するオフライン計算で 128 bit の安全性はない．ただし，文献 [11] で設計者が述べているように，この攻撃が Xoodyak の安全性に直接影響を及ぼすものではないことに注意されたい．

Xoodyak のハッシュ関数の安全性　Xoodyak のハッシュ関数利用モードは，Sponge 構造 [6] である．Sponge 構造は，b bit のランダム置換を用いる場合，b bit のうち r bit が入力メッセージの処理で用いられるレートと呼ばれるパラメータで，残りの c bit $(c = b - r)$ が安全性に寄与するキャパシティと呼ばれるパラメータである．Sponge 構造は，Indifferentiability の意味で，$c/2$ bit の安全性を持つ．Xoodyak のハッシュ関数は，$b = 384, c = 256$ で，Indifferentiability の安全性証明により，Xoodoo[12] をブラックボックスとする攻撃に対しては，Xoodyak は設計者が主張する安全性を持つ．Xoodoo[12] の構造を含めて考えると，Xoodoo[12] 単体での安全性は無いため，Sponge 構造の Indifferentiability の安全性は Xoodyak の安全性に適用できない．一方で，Xoodyak のハッシュ関数に対する攻撃は今のところ存在しないため，Xoodyak のハッシュ関数は設計者が主張する安全性を持つと考えられる．

Xoodyak の認証暗号の安全性　Xoodyak の認証暗号利用モードは，文献 [7, 12] に記載の置換をプリミティブとする Duplex 構造をベースに設計されている．Duplex 構造の安全性は文献 [12] で，ランダム関数との識別不可能性が示されており，Xoodyak の認証暗号の安全性は，Duplex 構造の安全性に依存する．Duplex 構造の安全性の結果に，Xoodyak のパラメータを適用すると，秘匿性に関しては，置換がブラックボックスの場合に，設計者が主張する安全性を持つと考えられる．一方で，偽造不可能性に関しては，文献 [12] の結果を用いると，置換がブラックボックスの場合に，オンライン計算で 64 bit，オフライン計算で 128 bit の安全性となるが，設計者が主張する安全性は，オンライン計算で 160 bit，オフライン計算で 128 bit と異なる．今のところ，設計者が主張する偽造不可能性を破る攻撃は存在しないが，安全性証明と著者の主張に差があることに注意が必要である．また，Xoodoo[12] の構造を含めて考えると，Xoodoo[12] 単体での安全性は無いため，Duplex 構造の安全性は Xoodyak の安全性に適用できない．Xoodoo[12] の構造を含めた Xoodyak の認証暗号に対する攻撃は，ラウンド数を 6 ラウンドに削減した Xoodyak の認証暗号に対する Conditional Cube 攻撃 [24] が最適であり，Xoodoo[12] のラウンド数は 12 ラウンドと十分にマージンがあるため，Xoodyak の認証暗号は設計者が主張する安全性を持つと考えられる．

目 次

1　本報告書の構成

本報告書の構成は以下の通りである．

- 2 章で記号の定義，プリミティブの定義，認証暗号に関する定義，ハッシュ関数に関する定義を説明する．
- 3 章で GIFT-COFB の仕様を記載する．
- 4 章で GIFT-COFB の安全性に関する文献調査の結果を報告する．
- 5 章で Xoodyak の仕様を記載する．
- 6 章で Xoodyak の安全性に関する文献調査の結果を報告する．
- 7 章で結論を述べる．

2　準備

本章では，記号の定義，GIFT-COFB と Xoodyak に必要なプリミティブの定義，認証暗号とハッシュ関数の定義を記載する．

2.1　記号の定義

本評価報告書では以下の記号を用いる．

- ε: 空列.
- $\emptyset$: 空集合.
- $[i, j]$:　$i \leq j$ となる整数 i, j に対し，i 以上 j 以下の整数の集合．$[i, j] = \{i, i+1, \ldots, j\}$ である．
- $(j] := [0, j]$，$[j] := [1, j]$.
- 0^i: 全てのビットが 0 の i bit の値.
- 1^i: 全てのビットが 1 の i bit の値.
- $\{0,1\}^i$: 全ての i bit の値の集合．i は非負の整数，$\{0,1\}^0 := \{\varepsilon\}$ とする．
- $\{0,1\}^*$: 全てのビット列の集合．$\{0,1\}^* = \cup_{i \in \{0,1,2,\ldots\}} \{0,1\}^i$ である．
- $\{0,1\}^{\leq i}$: ビット長が i 以下の全ての値の集合．i は非負の整数とする．
- $X \leftarrow Y$: X に値 Y を代入する．

- $T \xleftarrow{\$} \mathcal{T}$: 集合 $\mathcal{T}$ から値をランダムに選んで，その値を T に代入する．
- $|X|$：i bit の値 X のビット長．$|X| = i$ である．
- $X \oplus Y$: $|X| = |Y|$ の値 X と Y の XOR 演算の結果の値.
- $X \| Y$: 2 つのビット列 X と Y をビット結合した値.

2.2 プリミティブの定義

認証暗号は，ブロック暗号や置換など入出力長が固定のプリミティブをベースに設計される．Xoodyak のハッシュ関数と認証暗号は置換を用いて設計されており，GIFT-COFB の認証暗号はブロック暗号を用いて設計されている．ここでは，置換の定義，Xoodyak の利用モードの安全性に必要なランダム置換の定義，ブロック暗号の定義，そして GIFT-COFB の利用モードで用いるブロック暗号に対する安全性仮定である PseudoRandom Permutation(PRP) 安全性の定義を説明する．

2.2.1 置換に関する定義

置換のサイズを b bit とする．置換 $f : \{0,1\}^b \to \{0,1\}^b$ は，$\{0,1\}^b$ から $\{0,1\}^b$ への全単射関数である．

Xoodyak のモードの安全性を証明する際に，置換 f をランダム置換に置き換える．ランダム置換は，b bit の全ての置換の集合からランダムに 1 つ選んだものである．

2.2.2 ブロック暗号に関する定義

ブロック暗号の鍵サイズを k bit，ブロックサイズを n bit とする．ブロック暗号 $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ は，鍵成分を固定すると n bit の置換となり，別の鍵成分を用いると別の n bit の置換となる，2^k 個の n bit の置換を持つ関数である．鍵 $K \in \{0,1\}^k$ を用いるブロック暗号を E_K と書くことにする．

GIFT-COFB のモードの安全性を保証する際に，ブロック暗号 E_K を PRP 安全と仮定する．ここでは，E_K の PRP 安全性の定義を示す．PRP 安全性は，E_K とランダム関数 π の識別不可能性を考える．具体的には，攻撃者 A に対する以下のアドバンテージ関数が無視できるくらい小さいと仮定する．ここで，攻撃者 A はクエリー後 1 bit の値を出力する．そして，$\mathsf{A}^{\mathcal{O}}$ をオラクル $\mathcal{O}$ にアクセスする攻撃者 A の出力とする．

$$\mathsf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}) := \mathsf{Pr}\left[\mathsf{A}^{E_K} = 1\right] - \mathsf{Pr}\left[\mathsf{A}^{\pi} = 1\right] \ .$$

2.3　認証暗号に関する定義

認証暗号の関数一覧と機能を説明し，認証暗号の安全性である，AE 安全性，Priv 安全性，Auth 安全性を定義する．

2.3.1　認証暗号の関数一覧と機能

認証暗号は，データの秘匿機能と改ざん検知機能を持つ共通鍵暗号アルゴリズムである．認証暗号は，暗号化関数と復号関数を備える．認証暗号 Π の暗号化関数を $\Pi.\mathcal{E}$，復号関数を $\Pi.\mathcal{D}$ とするする．

認証暗号の暗号化関数 $\Pi.\mathcal{E}$ の入力は以下の 4 つである．

- 秘密鍵 K: 暗号化関数と復号関数で用いる秘密のデータ.
- ナンス N: 鍵が変わるまで暗号化関数の呼び出し毎に値が異なるデータで，改ざん検知対象となるデータ.
- Associated Data A: 秘匿はしないが改ざん検知対象となるデータ.
- 平文 M: 秘匿と改ざん検知対象となるデータ.

この入力に対し，$\Pi.\mathcal{E}$ の出力は以下の 2 つである．

- 暗号文 C: 平文 M の暗号化データ.
- タグ T: N，C，T の改ざんをチェックするためのデータ.

ここで，鍵 K を用いる認証暗号の暗号化関数 $\Pi.\mathcal{E}$ を $\Pi.\mathcal{E}_K$，入力 (K, N, A, M) の出力を $\Pi.\mathcal{E}_K(N, A, M)$ と書くことにする．$\Pi.\mathcal{E}_K(N, A, M) = (C, T)$ である．

認証暗号の復号関数 $\Pi.\mathcal{D}$ の入力は以下の 4 つである．

- 秘密鍵 K.
- ナンス N: 暗号化関数とは異なり，鍵を変更する前でも同じ値としてよく，改ざん検知対象となるデータ.
- Associated Data A: 秘匿はしないが，改ざん検知対象となるデータ.
- 暗号文 C: 平文の暗号化データで，改ざん検知対象となるデータ.
- タグ T': N，C，T' の改ざん検知を行うためのデータ.

この入力に対し，$\Pi.\mathcal{D}$ の出力は以下の 2 つである．

- 改ざんされていないことが確認できた場合，平文 M.
- 改ざんされていることが確認できた場合，改ざんがあったことを表す記号 reject.

ここで，鍵 K を用いる認証暗号の復号関数 $\Pi.\mathcal{D}$ を $\Pi.\mathcal{D}_K$，入力 (K, N, A, C, T') に対する出力を $\Pi.\mathcal{D}_K(N, A, C, T')$ と書くことにする．改ざんされていないことが確認できた場合，$\Pi.\mathcal{E}_K(N, A, C, T') = M$，改ざんされていることが確認できた場合，$\Pi.\mathcal{E}_K(N, A, C, T') = \mathsf{reject}$ である．

そして，認証暗号の暗号化関数 $\Pi.\mathcal{E}$ と復号関数 $\Pi.\mathcal{D}$ は，次の性質を持つ関数とする．

- $\Pi.\mathcal{E}$ の入力値の空間の任意の鍵 K，ナンス N，Associated Data A，暗号文 M に対し，$M = \Pi.\mathcal{D}_K(\Pi.\mathcal{E}_K(N, A, M))$ が成り立つ．

また，認証暗号の暗号化関数の入力のナンス N と，復号関数で計算される平文 M に関する条件を以下に記載する．

- 認証暗号の暗号化関数の入力のナンス N は，鍵を変えない限り同じ値を使わないことが必要である．この条件をナンスリスペクトと呼ぶ．
- 多くの認証暗号では，復号関数の計算で，改ざん検知のチェックを行う前に入力の暗号文を復号する．この，改ざん検知のチェック前の平文を Unverified Plaintext と呼ぶ．復号関数は改ざんされていないことが確認できない限り，Unverified Plaintext は出力しないことが必要である．

2.3.2 AE 安全性: 理想的な認証暗号との識別不可能性

認証暗号の安全性である AE 安全性は，理想的な認証暗号との識別不可能性で，文献 [18] で定義されている．以下，認証暗号 Π のシングルユーザでの AE 安全性を定義する．

理想的な認証暗号は，理想的な暗号化オラクルである $\$$ と理想的な復号オラクルである $\perp$ から構成される．これらのオラクルの説明は以下のとおりである．

- $\$$ は，$\Pi.\mathcal{E}_K$ と同じインターフェースを持ち，クエリー (N, A, M) に対して，ビット長が $|\Pi.\mathcal{E}_K(N, A, M)|$ の乱数を返す．
- $\perp$ は，$\Pi.\mathcal{D}_K$ と同じインターフェースを持ち，クエリー (N, A, C, T') に対して，常に reject を返す．

次に，認証暗号 Π の AE 安全性を破る攻撃者 A を説明する．攻撃者 A は，Real World と Ideal World で以下のオラクルにアクセスする．ここで，P は認証暗号で用いるプリミティブを理想化したものであり，認証暗号のプリミティブを理想化したモデルで出現する．例えば，Xoodyak のモードは，置換を理想化したランダム置換で安全性証明が行われており，その時の P はランダム置換である．一方で，COFB などスタンダードモデルで安全性証明を行う場合，P は無視する．

- Real World でのオラクルの組は，$(\Pi.\mathcal{E}_K, \Pi.\mathcal{D}_K, P)$ である．鍵 K は鍵空間からランダムに選ぶ．
- Ideal World でのオラクルの組は，$(\$, \perp, P)$ である．

そして，攻撃者 A は，全てのクエリーに対するレスポンスを受け取った後，1 bit の値を出力する．ここで，攻撃者 A は，Trivial なクエリーでなければ，自由に選択できる．Trivial なクエリーは，確率 1 で攻撃に成立するクエリーのことである．具体的には，暗号化関数 $\Pi.\mathcal{E}_K$ のレスポンスをそのまま復号関数 $\Pi.\mathcal{E}_K$ へのクエリーとするものを指す．また，ナンスリスペクトの状況を考える場合，$\Pi.\mathcal{E}_K/\$$ へのクエリーのナンスは全て異なる値である．

次に，AE 安全性の攻撃者 A のアドバンテージ関数を定義する．オラクルの組 $(\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3)$ へアクセスした場合の攻撃者 A の出力値を $\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_3}$ とする．AE 安全性の攻撃者 A のアドバンテージ関数の定義は以下の通りである．

$$\mathsf{Adv}^{\mathsf{ae}}_{\Pi}(\mathsf{A}) := \mathsf{Pr}\left[\mathsf{A}^{\Pi.\mathcal{E}_K,\Pi.\mathcal{D}_K,P} = 1\right] - \mathsf{Pr}\left[\mathsf{A}^{\$,\perp,P} = 1\right] \ .$$

ここで，スタンダードモデルでの安全性を考える場合，P を無視する．

2.3.3 Priv 安全性: 秘匿機能に関する安全性

Priv 安全性は，認証暗号の秘匿機能に関する安全性で，$\Pi.\mathcal{E}_K$ と理想的な暗号化オラクルである $\$$ との識別不可能性である．

認証暗号 Π の Priv 安全性を破る攻撃者 A を説明する．攻撃者 A は，Real World と Ideal World で以下のオラクルにアクセスする．P は，プリミティブを理想化したもので，プリミティブを理想化したモデルを考える場合，出現し，スタンダードモデルでの安全性を考える場合，無視する．

- Real World でのオラクルの組は，$(\Pi.\mathcal{E}_K, P)$ である．鍵 K は鍵空間からランダムに選ぶ．
- Ideal World でのオラクルの組は，$(\$, P)$ である．

そして，攻撃者 A は，全てのクエリーに対するレスポンスを受け取った後，1 bit の値を出力する．ここで，ナンスリスペクトの状況を考える場合，$\Pi.\mathcal{E}_K/\$$ へのクエリーのナンスは全て異なる値とする．

次に，Priv 安全性の攻撃者 A のアドバンテージ関数を定義する．オラクルの組 $(\mathcal{O}_1, \mathcal{O}_2)$ へアクセスした場合の攻撃者 A の出力値を $\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2}$ とすると，Priv 安全性の攻撃者 A のアドバンテージ関数の定義は以下の通りである．

$$\mathsf{Adv}^{\mathsf{priv}}_{\Pi}(\mathsf{A}) := \mathsf{Pr}\left[\mathsf{A}^{\Pi.\mathcal{E}_K,P} = 1\right] - \mathsf{Pr}\left[\mathsf{A}^{\$,P} = 1\right] \ .$$

ここで，スタンダードモデルでの安全性を考える場合，P を無視する．

AE 安全性と Priv 安全性の関係について，Priv 安全性は AE 安全性の復号オラクルを取り除いたものなので，認証暗号 Π が AE 安全であれば，Priv 安全性も担保される．そして，AE 安全性のアドバンテージ関数のバウンドから復号オラクルへのクエリーを除いたものが Priv 安全性のアドバンテージ関数のバウンドとなる．

2.3.4 Auth 安全性: 改ざん検知機能に関する安全性

Auth 安全性は，認証暗号の偽造不可能性に関する安全性である．

認証暗号 Π の Auth 安全性を破る攻撃者 A を説明する．攻撃者 A は，以下のオラクルにアクセスする．P は，プリミティブを理想化したもので，プリミティブを理想化したモデルを考える場合，出現し，スタンダードモデルでの安全性を考える場合，無視する．また，鍵 K は鍵空間からランダムに選ぶ．

- Π の暗号化関数 $\Pi.\mathcal{E}_K$，復号関数数 $\Pi.\mathcal{D}_K$，理想化したプリミティブ P.

攻撃者のゴールは，$\Pi.\mathcal{D}_K$ に，平文 (reject ではない値) がレスポンスとして返ってくる値をクエリーすることである．A は，各クエリーの値を，Trivial なクエリーでなければ，自由に選択できる．Trivial なクエリーは，確率 1 で攻撃に成立するクエリーのことで，暗号化関数 $\Pi.\mathcal{E}_K$ のレスポンスをそのまま復号関数 $\Pi.\mathcal{D}_K$ へのクエリーとするもを指す．また，ナンスリスペクトの状況を考える場合，各クエリーのナンスは全て異なる値である．

次に，Auth 安全性の攻撃者 A のアドバンテージ関数を定義する．攻撃者 A から $\Pi.\mathcal{D}_K$ へのあるクエリーに対するレスポンスが平文となるイベントを "$\mathsf{A}^{\Pi.\mathcal{E}_K,\Pi.\mathcal{D}_K,P}$ forges" と書くことにする．この時，Auth 安全性の攻撃者 A のアドバンテージ関数の定義は以下の通りである．

$$\mathsf{Adv}^{\mathsf{auth}}_{\Pi}(\mathsf{A}) := \mathsf{Pr}\left[\mathsf{A}^{\Pi.\mathcal{E}_K,\Pi.\mathcal{D}_K,P} \text{ forges}\right] \ .$$

ここで，スタンダードモデルでの証明の場合，P を無視する．

AE 安全性と Auth 安全性の関係について，認証暗号 Π の Auth 安全性を破る攻撃者が存在すれば，AE 安全性を破る攻撃者を構成できるため，認証暗号 Π が AE 安全であれば，Auth 安全性も担保される．そして，AE 安全性のアドバンテージ関数のバウンドが Auth 安全性のアドバンテージ関数のバウンドとなる．

2.4 ハッシュ関数に関する定義

ハッシュ関数は，任意長のデータを固定長のデータに変換する鍵なしの関数で，改ざん検知アルゴリズム，公開鍵暗号，デジタル署名など多くの暗号アルゴリズムの部品として用いられている．これらの暗号アルゴリズムの

安全性を担保するためには，ハッシュ関数が，Indifferentiability，Collision Resistance，Second Preimage Resistance，Preimage Resistance，m-Target Preimage Resistance などの安全性を持つ必要がある．以下，これらの安全性を定義する．

ここで，ハッシュ関数のプリミティブを P とする．P は，例えば置換などがある．プリミティブ P を用いる，出力長 ℓ bit のハッシュ関数を $H^P : \{0,1\}^* \to \{0,1\}^\ell$ とする．

2.4.1 Indifferentiability

Indifferentiability は，文献 [17] で定義された，ハッシュ関数と理想的なハッシュ関数であるランダムオラクル $\mathcal{F} : \{0,1\}^* \to \{0,1\}^\ell$ との識別不可能性である．Indifferentiability では，プリミティブ P を理想化する．P が置換の場合，ランダム置換を用いる．そして，ハッシュ関数 H^P が Indifferentiability の意味で安全ならば，文献 [20] で示された通りシングルステージの安全性ゲームに限定するが，ハッシュ関数 H^P をランダムオラクルとして用いることができる．すなわち，ランダムオラクルモデルで安全な暗号アルゴリズムに，Indifferentiability 安全なハッシュ関数を用いてもその安全性は担保される．また，Indifferentiability 安全なハッシュ関数は，Indifferentiability，Collision Resistance，Second Preimage Resistance，Preimage Resistance，m-Target Preimage Resistance に関して，安全である．

まず，Indifferentiability の攻撃者 A を説明する．攻撃者 A は，Real World と Ideal World で以下のオラクルにアクセスする．

- Real World でのオラクルの組は，(H^P, P) である．
- Ideal World でのオラクルの組は，$(\mathcal{F}, S)$ である．S は $\mathcal{F}$ にアクセス可能なシミュレータであり，プリミティブ P と同じインターフェースを持つ．

攻撃者 A は，オラクルへのクエリーに対するレスポンスを全て受け取った後，1 bit の値を出力する．

次に，Indifferentiability の攻撃者 A のアドバンテージ関数を定義する．オラクルの組 $(\mathcal{O}_1, \mathcal{O}_2)$ にアクセスする攻撃者 A の出力を $\mathsf{A}^{\mathcal{O}_1,\mathcal{O}_2}$ とすると，アドバンテージ関数は以下の通りである．

$$\mathsf{Adv}^{\mathsf{indiff}}_{H^P}(\mathsf{A}) := \mathsf{Pr}\left[\mathsf{A}^{H^P,P} = 1\right] - \mathsf{Pr}\left[\mathsf{A}^{\mathcal{F},S} = 1\right] \ .$$

そして，任意の攻撃者に対し，上記のアドバンテージ関数が無視できるくらい小さい確率となるシミュレータ S が存在する場合，そのハッシュ関数 H^P を Indifferentiability 安全とする．すなわち，ハッシュ関数 H^P の Indifferentiability 安全性を証明する場合，シミュレータ S を定義し，そのシミュレータに対してアドバンテージ関数のバウンドが無視できる確率となることを示す．

2.4.2 Collision Resistance

ハッシュ関数が Collision Resistance の意味で安全とは，$H^P(M) = H^P(M')$ となる異なる 2 つの入力メッセージ M, M' を見つけることが困難，具体的にはこの確率が無視できるくらい小さいことである．より詳しい定義は文献 [21] を参照されたい．

2.4.3 Second Preimage Resistance

ハッシュ関数が Second Preimage Resistance の意味で安全とは，入力メッセージ M が与えられた時に，$H^P(M) = H^P(M')$ となる M とは異なる入力メッセージ M' を見つけることが困難，具体的にはこの確率が無視できるくらい小さいことである．より詳しい定義は文献 [21] を参照されたい．

2.4.4 Preimage Resistance

ハッシュ関数が Preimage Resistance の意味で安全とは，入力メッセージ M のハッシュ値 $H^P(M)$ が与えられた時に，$H^P(M) = H^P(M')$ となる入力メッセージ M' を見つけることが困難，具体的にはこの確率が無視できるくらい小さいことである．より詳しい定義は文献 [21] を参照されたい．

2.4.5 m-Target Preimage Resistance

ハッシュ関数が Preimage Resistance の意味で安全とは，m 個の入力メッセージ $M_1, \ldots, M_m$ のハッシュ値 $H^P(M_1), \ldots, H^P(M_m)$ が与えられた時に，ある $i \in [m]$ に関して，$H^P(M_i) = H^P(M')$ となる入力メッセージ M' を見つけることが困難，具体的にはこの確率が無視できるくらい小さいことである．

3 GIFT-COFB のアルゴリズム仕様

本章で，GIFT-COFB のアルゴリズムの仕様を記載する．GIFT-COFB は，ブロック暗号ベースの認証暗号利用モード COFB とブロック暗号 GIFT-128 を組み合わせた方式である．以下，COFB の仕様を記載し，最後に GIFT-128 の仕様を記載する．詳細な仕様は文献 [2, 3] を参照されたい．

3.1 COFB の仕様

ブロック暗号ベースの認証暗号利用モード COFB の仕様を記載する．まず，COFB 記号とパラメータを定義する．次に，COFB で用いられるガロア

体，COFB で用いられる関数の仕様を記載する．最後に，COFB の暗号化関数 COFB.$\mathcal{E}$ と復号関数 COFB.$\mathcal{D}$ の仕様を記載する．

3.1.1 記号，パラメータの定義

COFB が用いるブロック暗号の記号は 2.2.2 章に記載のものを用いる．GIFT-128 は，$k = 128$, $n = 128$ である．COFB の鍵を K とする．COFB の鍵は，COFB で用いられるブロック暗号の鍵である．K のビット長は k bit である．

3.1.2 ガロア体の定義

COFB は $GF(2^{n/2})$ 上のかけ算と加算を用いる．COFB では，$GF(2^{n/2})$ 上のかけ算または加算を行う際に，$n/2$ bit の値を $GF(2^{n/2})$ 上の値に変換し演算を行う．そして，演算後，演算結果を $n/2$ bit の値に変換する．$a_0, a_1, \cdots, a_{n/2-1} \in \{0,1\}$ に対して，$n/2$ bit の値 $a_{n/2-1} \ldots a_0$ に対して $GF(2^{n/2})$ 上の演算を行う場合，$GF(2^{n/2})$ 上の値 (多項式)$a_{n/2-1}x^{n-1} + \cdots + a_1 x + a_0$ に変換する．$GF(2^{n/2})$ 上の値から $n/2$ bit の値に変換する際は，この逆の変換を行う．

$GF(2^{n/2})$ 上のかけ算を行う際は，α を生成元とする原始多項式 $p(x)$ を 1 つ固定し，かけ算を行う．α をビット表現すると 10 となる．GIFT-COFB では，以下の 64 次の原始多項式が用いられる．

$$p(x) = x^{64} + x^4 + x^3 + x + 1$$

COFB では生成元 α $(= 10)$ と 11 のかけ算が用いられる．以下，10 を整数 2 で表現し，11 を整数 3 で表現する．

$GF(2^{n/2})$ 上の値 X に，2 を i 回掛ける場合 $2^i X$，3 を i 回掛ける場合 $3^i X$ と書くことにする．

3.1.3 COFB で用いられる関数

COFB で用いられる 3 つの関数 $\mathrm{Trunc}, \mathrm{Pad}, G$ の仕様を以下に示す．

- Trunc_i は，n bit の値 X を入力とし，X の最初の i bit を出力する関数である．$i \leq n$ であり，この i bit の値を $\mathrm{Trunc}_i(X)$ と書くことにする．また，i は自然数であり，$i \leq n$ である．

- Pad は，n bit 以下の値 X を入力とし，$|X| = n$ の場合 X を出力し，$|X| < n$ の場合 $X \| 10^{n-|X|-1}$ を出力する関数である．この出力値を $\mathrm{Pad}(X)$ とする．

- G は，n bit の値 Y を入力とし，Y を上位 $n/2$ bit $Y[1]$，下位 $n/2$ bit $Y[2]$ に分割し，$Y[2]\|(Y[1] \lll 1)$ を出力する関数である．この出力値を $G(Y)$ と書くことにする．$Y = Y[1]\|Y[2]$ であり，$(Y[1] \lll 1)$ は $Y[1]$ を左に 1 bit 巡回シフトした値である．

3.1.4 COFB の暗号化関数 COFB.$\mathcal{E}$

COFB の暗号化関数 COFB.$\mathcal{E}$ の入出力は以下の通りである．t はタグ長で，n 以下の自然数である．

入力

- 鍵 $K \in \{0,1\}^k$
- ナンス $N \in \{0,1\}^n$
- Associated Data $A \in \{0,1\}^*$
- 平文 $M \in \{0,1\}^*$

出力

- 暗号文 $C \in \{0,1\}^{|M|}$
- タグ $T \in \{0,1\}^t$

COFB の暗号化関数 COFB.$\mathcal{E}$ の手続きは以下の通りである．

1. Associated Data A を n bit のデータブロック $A[1], A[2], \ldots, A[a]$ に分割する．ここで，$|A[1]| = \cdots = |A[a-1]| = n$，$|A[a]| \leq n$ であり，$A = A[1]\|A[2]\|\cdots\|A[a]$ である．
2. 図 1(1) を実行する．ここでは，ナンス N と Associated Data ブロック $A[1], A[2], \ldots, A[a]$ を処理し，$L \in \{0,1\}^{n/2}$ と $Y[a] \in \{0,1\}^n$ を生成する．
3. 平文 M を n bit のデータブロック $M[1], M[2], \ldots, M[m]$ に分割する．ここで，$|M[1]| = \cdots = |M[m-1]| = n$，$|M[m]| \leq n$ であり，$M = M[1]\|M[2]\|\cdots\|M[m]$ である．
4. $m \geq 2$ の場合，図 1(2) を実行する．ここでは，平文ブロック $M[1], \cdots, M[m-1] \in \{0,1\}^n$ を暗号化し，暗号文ブロック $C[1], \cdots, C[m-1] \in \{0,1\}^n$ と $Y[a+m-1] \in \{0,1\}^n$ を生成する．
5. $M \neq \varepsilon$ の場合，図 1(3) を実行する．ここでは，平文ブロック $M[m] \in \{0,1\}^{\leq n}$ を暗号化し，暗号文ブロック $C[m] \in \{0,1\}^{|M[m]|}$ とタグ $T \in \{0,1\}^t$ を生成し，$C = C[1]\|\cdots\|C[m]$ とする．

6. $M = \varepsilon$ の場合，図 1(4) を実行し，タグ $T \in \{0,1\}^t$ を生成する．$C = \varepsilon$ とする．

3.1.5 COFB の復号 COFB.$\mathcal{D}$

COFB の復号関数 COFB.$\mathcal{D}$ の入出力は以下の通りである．t はタグ長で，n 以下の自然数である．

入力

- 鍵 $K \in \{0,1\}^k$
- Associated Data $A \in \{0,1\}^*$
- 暗号文 $C \in \{0,1\}^{|M|}$
- タグ $T' \in \{0,1\}^t$

出力

- 暗号文 $M \in \{0,1\}^{|C|}$，または，タグの改ざんを示す記号 $\perp \notin \{0,1\}^{|C|}$

COFB の復号関数 COFB.$\mathcal{D}$ の手続きは以下のとおりである．ここで，COFB.$\mathcal{D}$ の出力値を COFB.$\mathcal{D}_K(N, A, C, T')$ とする．

1. Associated Data A を n bit のデータブロック $A[1], A[2], \ldots, A[a]$ に分割する．ここで，$|A[1]| = \cdots = |A[a-1]| = n$，$|A[a]| \leq n$ であり，$A = A[1] \| A[2] \| \cdots \| A[a]$ である．
2. 図 2(1) を実行する．ここでは，ナンス N と Associated Data ブロック $A[1], A[2], \ldots, A[a]$ を処理し，$L \in \{0,1\}^{n/2}$ と $Y[a] \in \{0,1\}^n$ を生成する．
3. 暗号文 C を n bit のデータブロック $C[1], M[2], \ldots, C[c]$ に分割する．ここで，$|C[1]| = \cdots = |C[c-1]| = n$，$|C[c]| \leq n$ であり，$C = C[1] \| C[2] \| \cdots \| C[c]$ である．
4. $c \geq 2$ の場合，図 2(2) を実行する．ここでは，暗号文ブロック $C[1], \cdots, C[c-1] \in \{0,1\}^n$ を復号し，平文ブロック $M[1], \cdots, M[c-1] \in \{0,1\}^n$ と $Y[a+c-1] \in \{0,1\}^n$ を生成する．
5. $C \neq \varepsilon$ の場合，図 2(3) を実行する．ここでは，暗号文ブロック $C[c] \in \{0,1\}^{\leq n}$ を復号し，平文ブロック $M[c] \in \{0,1\}^{|M[c]|}$ とタグ $T \in \{0,1\}^t$ を生成し，$M = M[1] \| \cdots \| M[c]$ とする．

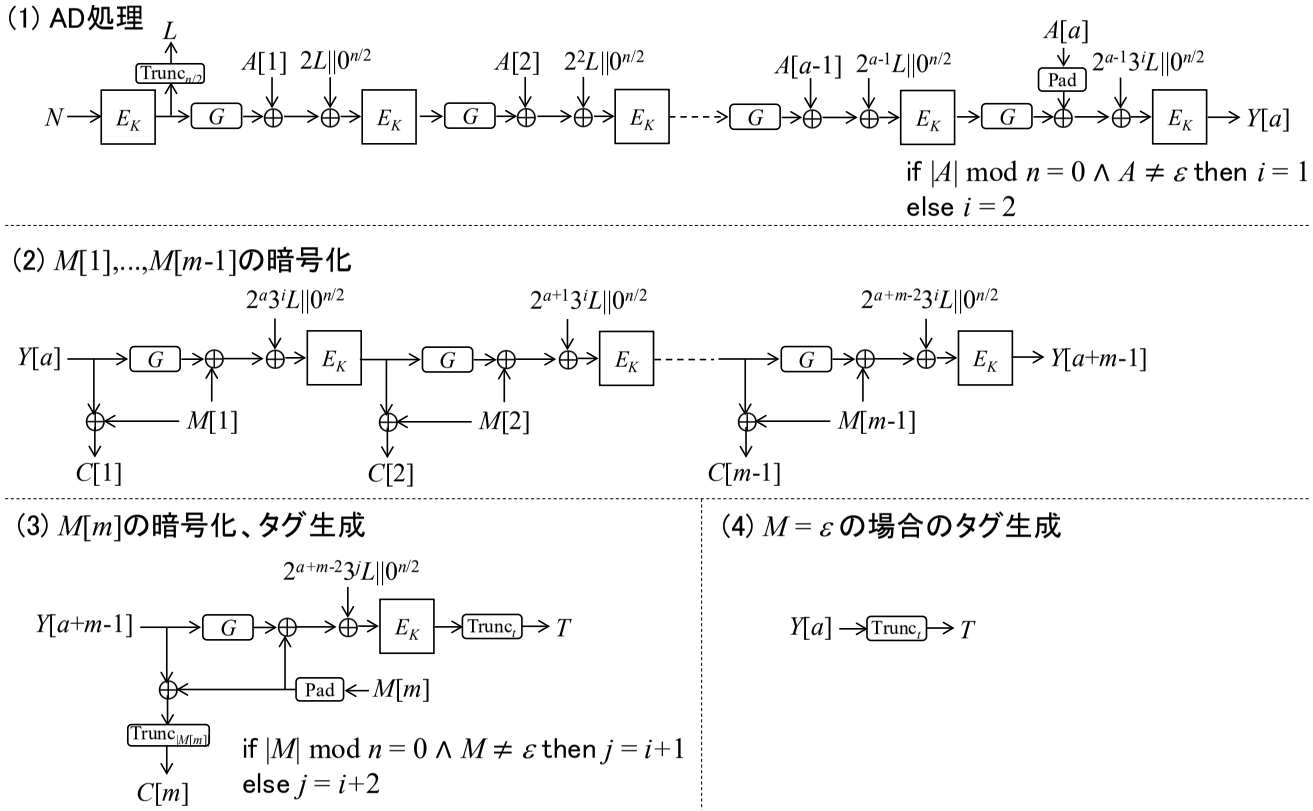


図 1: COFB の暗号化関数 COFB.$\mathcal{E}$ の手続き

6. $C = \varepsilon$ の場合，図 2(4) を実行し，タグ $T \in \{0,1\}^t$ を生成し，$M = \varepsilon$ とする．
7. $T = T'$ の場合，$\mathsf{COFB}.\mathcal{D}_K(N, A, C, T') = M$ とする．
8. $T \neq T'$ の場合，$\mathsf{COFB}.\mathcal{D}_K(N, A, C, T') = \mathsf{reject}$ とする．

3.2 GIFT-128 の仕様

GIFT-128 の仕様を記載する．GIFT-COFB は GIFT-128 の暗号化関数のみ用いるため，ここでは，暗号化関数の仕様を記載する．

GIFT-128 は，鍵サイズとブロックサイズがともに 128 bit のブロック暗号であり，ラウンド関数を 40 回繰り返す構造を持つ．128 bit の平文ブロックを 128 bit のステート S の初期値とし，40 回ラウンド関数を繰り返しステート S をアップデートする．40 ラウンド後の S の値が GIFT-128 の出力の暗号文ブロックとなる．各ラウンド関数では，鍵から GIFT-128 の鍵スケジュールで生成されたラウンド鍵が用いられる．以下に，ラウンド関数と鍵スケジュールの仕様を示す．

3.2.1 鍵スケジュールの仕様

初期化 まず，鍵ステート KS の初期値を記載する．鍵の先頭から i bit 目の値を b_{i-1} とする．すなわち，$b_0 \| b_1 \| \cdots \| b_{127}$ は鍵である．そして，鍵ステートの初期値は以下のように決まる．$W_0, W_1, \ldots, W_7$ の各ワードは 16 bit の値である．

$$KS = \begin{bmatrix} W_0 & \| & W_1 \\ W_2 & \| & W_3 \\ W_4 & \| & W_5 \\ W_6 & \| & W_7 \end{bmatrix} \leftarrow \begin{bmatrix} b_{127} & \cdots & b_{112} & \| & b_{111} & \cdots & b_{98} & b_{97} & b_{96} \\ b_{95} & \cdots & b_{80} & \| & b_{79} & \cdots & b_{66} & b_{65} & b_{64} \\ b_{63} & \cdots & b_{48} & \| & b_{47} & \cdots & b_{34} & b_{33} & b_{32} \\ b_{31} & \cdots & b_{16} & \| & b_{15} & \cdots & b_2 & b_1 & b_0 \end{bmatrix}$$

鍵スケジュールの手続き 鍵スケジュールでは，各ラウンドの 64bit のラウンド鍵 RK を鍵ステート KS から生成したのち，鍵ステートをアップデートする．

まず，ラウンド鍵の生成法を以下に示す．

$$U \leftarrow W_2 \| W_3, \ \ V \leftarrow W_6 \| W_7, \ \ RK \leftarrow U \| V$$

次に，鍵ステートのアップデート方法を以下に示す．

$$\begin{bmatrix} W_0 & \| & W_1 \\ W_2 & \| & W_3 \\ W_4 & \| & W_5 \\ W_6 & \| & W_7 \end{bmatrix} \leftarrow \begin{bmatrix} W_6 \ggg 2 & \| & W_{17} \ggg 12 \\ W_0 & \| & W_1 \\ W_2 & \| & W_3 \\ W_4 & \| & W_5 \end{bmatrix}$$

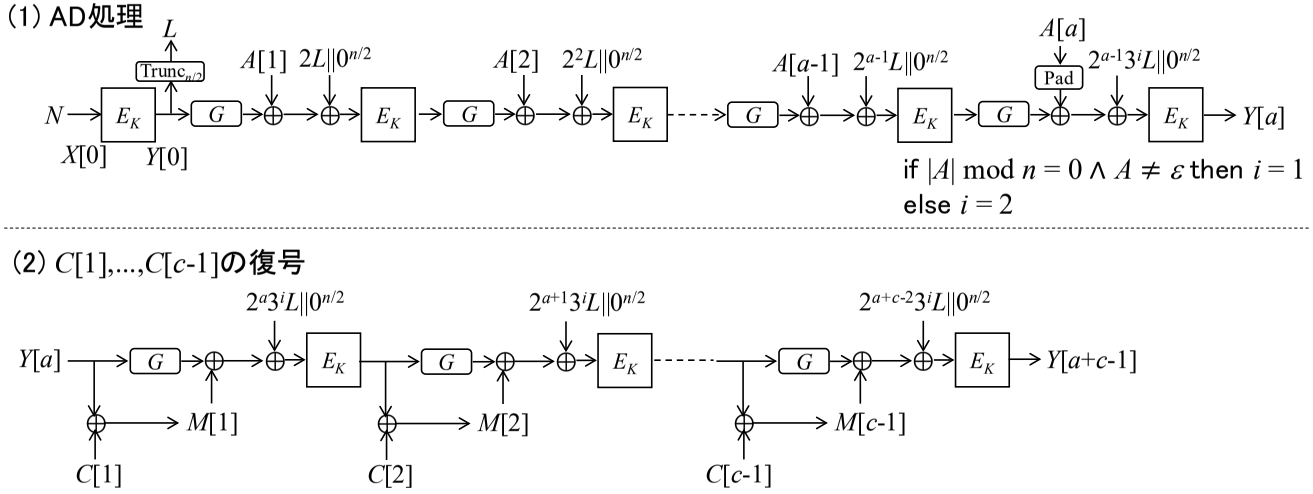


図 2: COFB の暗号化関数 COFB.$\mathcal{D}$ の手続き

ここで，1 ワード W に対し，$W \ggg i$ は W を $i$bit 右巡回シフトした値である．r ラウンドで用いられるラウンド鍵 RK は，上記のアップデートを $r-1$ 回繰り返した後の鍵ステート KS から生成された値である．

3.2.2 ラウンド関数の仕様

初期化 ステート S の初期値を記載する．平文ブロックの先頭から i bit 目の値を b_{i-1} とする．すなわち，$b_0 \| b_1 \| \cdots \| b_{127}$ は平文ブロックである．そして，ステートの初期値は以下のように定義される．S_0, S_1, S_2, S_3 の各ワードは 32bit の値である．

$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \leftarrow \begin{bmatrix} b_{124} & \cdots & b_8 & b_4 & b_0 \\ b_{125} & \cdots & b_9 & b_5 & b_1 \\ b_{126} & \cdots & b_{10} & b_6 & b_2 \\ b_{127} & \cdots & b_{11} & b_7 & b_3 \end{bmatrix}$$

ラウンド関数の手続き ラウンド関数は SubCells，PermBits，AddRoundKey の 3 ステップから構成される．SubCells，PermBits，AddRoundKey の順番で実行し，ステート S をアップデートする．

SubCells の手続きは以下のとおりである．各処理は 32bit ワード単位で行われている．

$$\begin{aligned} S_1 &\leftarrow S_1 \oplus (S_0 \& S_2) \\ S_0 &\leftarrow S_0 \oplus (S_1 \& S_3) \\ S_2 &\leftarrow S_2 \oplus (S_0 \mid S_1) \\ S_3 &\leftarrow S_3 \oplus S_2 \\ S_1 &\leftarrow S_1 \oplus S_3 \\ S_3 &\leftarrow \sim S_3 \\ S_2 &\leftarrow S_2 \oplus (S_0 \& S_1) \\ \{S_0, S_1, S_2, S_3\} &\leftarrow \{S_3, S_1, S_2, S_0\} \end{aligned}$$

ここで，& は AND 演算，| は OR 演算，$\sim$ は NOT 演算である．

PermBits の手続きは，S_0, S_1, S_2, S_3 の各ワードに対し，表 1 に記載のビット置換を行う．表 1 の Index の列は各ワード S_i のビット位置を示しており，S_1, S_2, S_3, S_4 の各列は，Index の列に記載のビット位置を移動した先のビット位置を示している．例えば，S_0 の 31bit 目の値は，PermBits 適用後 S_0 の 29bit 目となる．なお，各ワードの先頭のビット位置を 0，最後のビット位置は 31 である．

AddRoundKey の手続きは以下の通りである．

表 1: PermBits のビット置換表

Index	31	30	29	28	27	26	25	24	23	22	21
S_0	29	25	21	17	13	9	5	1	30	26	22
S_1	30	26	22	18	14	10	6	2	31	27	23
S_2	31	27	23	19	15	11	7	3	28	24	20
S_2	28	24	20	16	12	8	4	0	29	25	21

Index	20	19	18	17	16	15	14	13	12	11	10
S_0	18	14	10	6	2	31	27	23	19	15	11
S_1	19	15	11	7	3	28	24	20	16	12	8
S_2	16	12	8	4	0	29	25	21	17	13	9
S_2	17	13	9	5	1	30	26	22	18	14	10

Index	9	8	7	6	5	4	3	2	1	0
S_0	7	3	28	24	20	16	12	8	4	0
S_1	4	0	29	25	21	17	13	9	5	1
S_2	5	1	30	26	22	18	14	10	6	2
S_2	6	2	31	27	23	19	15	11	7	3

- まず，ラウンド鍵 RK を 2 つの 32bit の値 U と V に分ける．$RK = U\|V$ である．
- 次に，以下のように S_1, S_2, S_3 をアップデートする．

$$S_1 \leftarrow S_1 \oplus V$$
$$S_2 \leftarrow S_2 \oplus U$$
$$S_3 \leftarrow S_3 \oplus \texttt{0x800000XY}$$

ここで，`XY` は 8bit の値 $00c_5c_4c_3c_2c_1c_0$ であり，6bit の値 $c_5c_4c_3c_2c_1c_0$ は以下の 6-bit LFSR で更新される．

$$c_5\|c_4\|c_3\|c_2\|c_1\|c_0 \leftarrow c_4\|c_3\|c_2\|c_1\|c_0\|(c_5 \oplus c_4 \oplus 1)$$

各ビット c_i の初期値は 0 であり，r ラウンド目 $c_5\|c_4\|c_3\|c_2\|c_1\|c_0$ は，上記の LFSR を r 回適用した値である．

4 GIFT-COFB の安全性に関する文献調査の結果

GIFT-COFB の安全性は，GIFT-COFB の利用モードである COFB の安全性評価と，プリミティブ GIFT-128 の安全性評価をそれぞれ行い，2 つの評価

結果を組み合わせることで，示される．COFB の安全性評価では，ブロック暗号を PRP 安全と仮定し，COFB が AE 安全性を評価する．GIFT-128 の安全性評価では，GIFT-128 が PRP 安全となることを評価する．以下，GIFT-128 と GIFT-COFB(すなわち，COFB) の安全性に関する文献調の結果を報告する．

4.1 GIFT-128 の安全性評価に関する文献調査結果

GIFT-128 に対する解析論文はいくつか発表されている．調査時点で，攻撃可能なラウンド数の最大値は 27 ラウンドで，差分攻撃を用いた鍵復元である．GIFT-128 の PRP 安全性を破る攻撃法はなく，PRP 安全性を満たすと考えられる．以下，差分攻撃，線形攻撃，Integral 攻撃に対する攻撃論文の調査結果を記載する．

4.1.1 GIFT-128 に対する差分攻撃

文献 [25] は，MILP を用いて 20 ラウンドの GIFT-128 に対する確率 $2^{-121.813}$ の Distinguisher を提案した．そして，この Distinguisher を用いて，27 ラウンドの GIFT-128 に対する鍵復元攻撃を行っている．鍵復元に必要な時間，データ量，メモリーは，それぞれ $(2^{124.83}, 2^{123.53}, 2^{80})$ である．

4.1.2 GIFT-128 に対する線形攻撃

文献 [22, 23] は，SAT を用いて 19 ラウンドの GIFT-128 に対する確率 $2^{-123.11}$ の Distinguisher を提案した．そして，この Distinguisher を用いて，25 ラウンドの GIFT-128 に対する鍵復元攻撃を行っている．鍵復元に必要な時間，データ量，メモリーは，$(2^{126.77}, 2^{124.75}, 2^{96})$，または $(2^{127.77}, 2^{125.75}, 2^{96})$ である．

4.1.3 GIFT-128 に対する Integral 攻撃

GIFT-128 の設計者により 11 ラウンドの Distinguisher が発見されている [5]．

4.2 GIFT-COFB の安全性評価に関する文献調査結果

上記の通り，調査時点で，GIFT-128 は PRP 安全であり，GIFT-COFB の安全性は COFB の安全性に依存する．以下，COFB に関する文献調査の結果を報告する．

4.2.1 文献 [2] の COFB の安全性

文献 [2] で，以下の COFB の AE 安全性バウンドが示されている．

Theorem 1 (COFB の AE 安全性)**.** COFB の AE 安全性を破るナンスリスペクトの任意の攻撃者 A に対し，以下の不等式を満たす E_K に対する PRP 攻撃者 A' が存在する．ここで，A の動作時間を t とし，クエリー回数を q，復号関数へのクエリー回数を q_f，全てのクエリーのブロック長の和を σ，全ての復号関数へのクエリーのブロック長の和を σ_f，$q' = q + q_f + \sigma + \sigma_f$ とする．

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathrm{COFB}}(\mathsf{A}) \leq \mathsf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}') + \frac{\binom{q'}{2}}{2^n} + \frac{\sigma+1}{2^{n/2}} + \frac{q_f(n+4)}{2^{n/2+1}} + \frac{3\sigma^2 + q_f + 2(q+\sigma+\sigma_f)\cdot\sigma_f}{2^n} \ .$$

また，A' の動作時間は $t + O(q')$ であり，クエリー数は q' である．■

このバウンドはおよそ $2^{n/2}/n$ 回のクエリーでコンスタントの値になるため，COFB は $n/2 - \log_2 n$bit の AE 安全性を持つ．

4.2.2 設計者が主張する GIFT-COFB の安全性について

COFB の秘匿に関する安全性である Priv 安全性について，AE 安全性の定義で $q_f = 0$，$\sigma_f = 0$ とすると，AE 安全性は Priv 安全性の定義そのものになるため，上記の AE 安全性のバウンドから COFB の秘匿に関する安全性である Priv 安全性のバウンドを得ることができる．上記のバウンドから，COFB がオンラインクエリに関して $n/2$ bit の Priv 安全性を持つことが言える．そして，GIFT-COFB のパラメータ $n = 128$，$k = 128$ を用いると，文献 [2] で GIFT-COFB の設計者が主張する，オンラインクエリに関して 64 bit，オフラインクエリに関して 112 bit 以上の Priv 安全性を持つことが言える．

COFB の改ざん検知に関する安全性である Auth 安全性について，AE 安全性の定義より，COFB の Auth 安全性を破る攻撃者が存在すれば，AE 安全を破る攻撃者が構成できるため，上記の AE 安全性のバウンドは，COFB の Auth 安全性のバウンドとなる．よって，COFB は $n/2 - \log_2 n$ bit の Auth 安全性を持つことが言える．GIFT-COFB のパラメータである $n = 128$，$k = 128$ を用いると，文献 [2] で GIFT-COFB の設計者が主張する，オンラインクエリに関して 58 bit，オフラインクエリに関して 112 bit 以上の Priv 安全性を持つことが言える．

4.2.3 文献 [9] の COFB の安全性バウンドの誤りについて

文献 [9] では，AE 安全性について以下のことが主張されている．

Claim 1 (文献 [9] の定理 2)**.** COFB の AE 安全性を破る任意のナンスリスペクトの攻撃者 A に対し，以下の不等式成り立つ E_K に対する PRP 攻撃者 A' が存在する．ここで，A の動作時間を t とし，暗号化関数へのクエリー回数を q_e，復号関数へのクエリー回数を q_f，全ての暗号化関数へのクエリーのブロック長の和を σ_e，全ての復号関数へのクエリーのブロック長の和を σ_f，$q' = q_e + q_f + \sigma_e + \sigma_f$ とする．

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathrm{COFB}}(\mathsf{A}) \leq \mathsf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}') + \frac{0.5(q')^2}{2^n} + \frac{\sigma_e}{2^{n/2}} + \frac{(q_e + \sigma_e + 2\sigma_f)\cdot\sigma_f + q_f}{2^n} \ .$$

また，A' の動作時間は $t + O(q')$ であり，クエリー数は q' である．■

しかし，文献 [15] で，文献 [9] が主張する上記のバウンドが間違いであることが示されている．文献 [15] では，$q_e = 1$ で，以下の確率で成功する COFB への偽造攻撃が示されている．

$$\frac{q_d}{2^{n/2}} \ .$$

この結果は，$q_e = 1$，$q_d = 2^{n/2}$ で，COFB をコンスタントの確率で破ることができることを示しているが，文献 [9] のバウンドはコンスタントにならないため，文献 [15] の偽造攻撃と矛盾する．これは，文献 [9] のバウンドが間違っていることを意味する．

4.2.4 文献 [3] の COFB の安全性バウンドの誤りについて

文献 [3] では，AE 安全性について以下のことが主張されている．

Claim 2 (文献 [3] の COFB の安全性バウンド)**.** ナンスリスペクトの COFB の AE 安全性を破る任意の攻撃者 A に対し，以下の不等式成り立つ E_K に対する PRP 攻撃者 A' が存在する．ここで，A の動作時間を t とし，暗号化関数へのクエリー回数を q_e，復号関数へのクエリー回数を q_f，全ての暗号化関数へのクエリーのブロック長の和を σ_e，全ての復号関数へのクエリーのブロック長の和を σ_f，$q' = q_e + q_f + \sigma_e + \sigma_f$ とする．

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathrm{COFB}}(\mathsf{A}) \leq \mathsf{Adv}^{\mathsf{prp}}_{E}(\mathsf{A}') + \frac{\binom{q}{2}}{2^n} + \frac{1}{2^{n/2}} + \frac{q_f(n+4)}{2^{2n/2+1}} + \frac{3\sigma + q_f + 2(q + \sigma + \sigma_f)\cdot\sigma_f}{2^n} \ .$$

また，A' の動作時間は $t + O(q')$ であり，クエリー数は q' である．■

しかし，文献 [14] で，文献 [3] が主張する上記のバウンドが間違いであることが示されている．文献 [14] では，以下の確率で成功する COFB の AE 安全性を破る攻撃が示されている．なお，この攻撃では，$q_f = 0$ である．

$$\frac{q_e}{2^{n/2}} \ .$$

例えば，$q_e = 2^{n/4}$，$q_f = 0$ の場合，この攻撃成功確率は $1/2^{n/4}$ となるが，上記の文献 [3] のバウンドは $1/2^{n/4}$ よりも小さいため，この攻撃と矛盾する．これは，文献 [3] のバウンドが間違っていることを意味する．

5 Xoodyak のアルゴリズム仕様

本章で，Xoodyak のアルゴリズムの仕様を記載する．Xoodyak は，置換ベースのハッシュ関数と，認証暗号を含む鍵付き関数から構成される．本報告書では，ハッシュ関数と認証暗号の仕様を示す．まず，ハッシュ関数の利用モードの仕様を記載する．次に，認証暗号の利用モードを記載する．最後に，Xoodyak の置換 Xoodoo[12] の仕様を記載する．また，詳細な仕様は文献 [11] を参照されたい．

5.1 利用モードの仕様

Xoodyak がサポートするハッシュ関数利用モードと認証暗号利用モードの仕様を記載する．

5.1.1 記号，パラメータの定義

f を Xoodyak で用いる b bit の置換とする．Xoodyak のハッシュ関数利用モードのブロックサイズを R_{hash}，Xoodyak の認証暗号利用モードの 2 つのブロックサイズを R_{kin}，R_{kout} とする．R_{kin} は，鍵，ナンス，AD を処理する際に使うパラメータで，R_{kout} は，平文，暗号文，タグを処理する際に使うパラメータである．ℓ をハッシュ関数の出力サイズ，k を鍵サイズ，n をナンスのサイズ，t をタグサイズとする．ただし，$k + n + 16 \leq R_{\mathrm{kin}}$ とする．また，各サイズはビット長で，8 の倍数とする．(0xVW) を，1 byte 値 0xVW を 8bit 表現した値とする．V と W は 16 進数の値である．$(0\mathrm{x}01) = 00000001$，$(0\mathrm{x}03) = 00000011$，$(0\mathrm{x}40) = 01000000$，$(0\mathrm{x}81) = 10000001$ である．なお，Xoodyak では，$R_{\mathrm{hash}} = 128$，$R_{\mathrm{kin}} = 352$，$R_{\mathrm{kout}} = 192$，$b = 384$，$\ell = 256$，$k = 128$，$t = 128$，$n = 128$ である．

5.1.2 Xoodyak のハッシュ関数利用モード

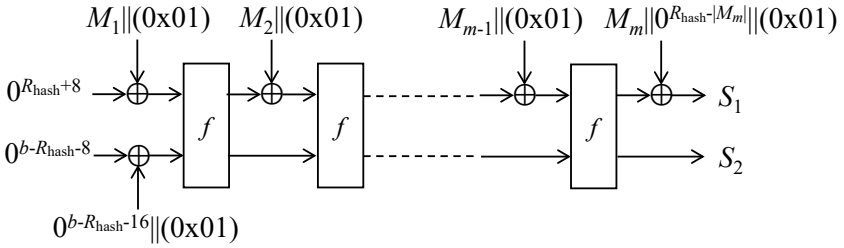
Xoodyak のハッシュ関数利用モードの入出力は以下の通りである．

入力

- メッセージ $M \in \{0,1\}^*$

出力

(1) メッセージの処理



(2) ハッシュ値の生成

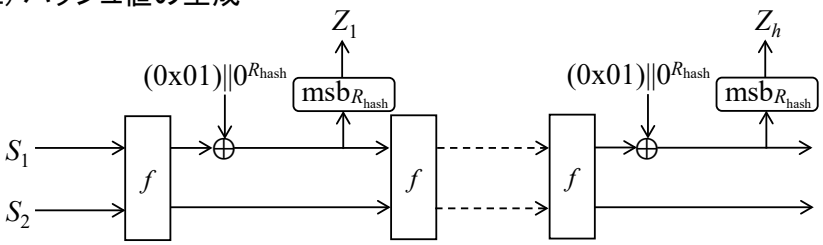


図 3: Xoodyak のハッシュ関数利用モードの手続き

- ハッシュ値 $Z \in \{0,1\}^{\ell}$

Xoodyak のハッシュ関数利用モードの手続きは以下の通りである．

1. 入力のメッセージ M を R_{hash} bit のメッセージブロック $M_1, M_2, \ldots, M_m$ に分割する．ここで，$|M_1| = \cdots = |M_{m-1}]| = R_{\text{hash}}$，$|M_m| \leq R_{\text{hash}}$ であり，$M = M_1 \| M_2 \| \cdots \| M_m$，$m = \lceil |M| / R_{\text{hash}} \rceil$ である．
2. 図 3(1) を実行する．ここでは，メッセージブロック $M_1, M_2, \ldots, M_m$ を処理し，$S_1 \in \{0,1\}^{R_{\text{hash}}+8}$ と $S_2 \in \{0,1\}^{b-R_{\text{hash}}-8}$ を生成する．
3. 図 3(2) を実行する．ここでは，R_{hash} bit の出力ブロック $Z_1, \ldots, Z_h$ を生成する．$h = \lceil \ell / R_{\text{hash}} \rceil$ である．
4. ハッシュ値を $Z = \mathsf{msb}_{\ell}(Z_1 \| \cdots \| Z_h)$ とする．

5.2 Xoodyak の認証暗号利用モードの仕様

Xoodyak の認証暗号利用モードは暗号化関数 Xoodyak.$\mathcal{E}$ と復号関数 Xoodyak.$\mathcal{D}$ から構成される．以下，これらの関数の仕様を記載する．

5.2.1 Xoodyak の認証暗号利用モードの暗号化関数 Xoodyak.$\mathcal{E}$

Xoodyak の認証暗号利用モードの暗号化関数 Xoodyak.$\mathcal{E}$ の入出力は以下の通りである．

入力

- 鍵 $K \in \{0,1\}^k$
- ナンス $N \in \{0,1\}^n$
- Associated Data $A \in \{0,1\}^*$
- 平文 $M \in \{0,1\}^*$

出力

- 暗号文 $C \in \{0,1\}^{|M|}$
- タグ $T \in \{0,1\}^t$

Xoodyak の認証暗号利用モードの暗号化関数の手続きは以下の通りである．

1. 図 4(1) を実行し，鍵 K，ナンス N から初期値 $S_{K,N}$ を生成する．
2. Associated Data A を R_{kin} bit のデータブロック $A_1, A_2, \ldots, A_a$ に分割する．ここで，$|A_1| = \cdots = |A_{a-1}| = R_{\mathrm{kin}}$，$|A_a| \leq R_{\mathrm{kin}}$ であり，$A = A_1 \| A_2 \| \cdots \| A_a$，$a = \lceil |A|/R_{\mathrm{kin}} \rceil$ である．
3. 図 4(2) を実行する．ここでは，Associated Data ブロック $A_1, A_2, \ldots, A_a$ を処理し，$S_{A,1}$ と $S_{A,2}$ を生成する．$|S_{A,1}| = R_{\mathrm{kin}} + 8$ と $|S_{A,2}| = b - (R_{\mathrm{kin}} + 8)$ であり，1 回目の f を計算して以降の 2 本の並列の線のうち，上の線は $R_{\mathrm{kin}} + 8$ bit，下の線は $b - (R_{\mathrm{kin}} + 8)$ bit である．
4. 平文 M を R_{kout} bit のデータブロック $M_1, M_2, \ldots, M_m$ に分割する．ここで，$|M_1| = \cdots = |M_{m-1}| = R_{\mathrm{kout}}$，$|M_m| \leq R_{\mathrm{kout}}$ であり，$M = M_1 \| M_2 \| \cdots \| M_m$，$m = \lceil |M|/R_{\mathrm{kout}} \rceil$ である．
5. 図 4(3) を実行する．ここでは，平文ブロック $M_1, \cdots, M_m$ を暗号化し，暗号文ブロック $C_1, \cdots, C_m$ と，$S_{M,1}$ と $S_{M,2}$ を生成する．各 $i \in [m]$ に対し，$|M_i| = |C_i|$，$|S_{M,1}| = R_{\mathrm{kout}} + 8$，$|S_{M,2}| = b - (R_{\mathrm{kout}} + 8)$ であり，1 回目の f を計算して以降の 2 本の並列の線のうち，上の線は $R_{\mathrm{kout}} + 8$ bit，下の線は $b - (R_{\mathrm{kout}} + 8)$ bit である．
6. 図 4(4) を実行する．ここでは，出力ブロック $Z_1, \ldots, Z_d$ を生成する．$d = \lceil t/R_{\mathrm{kout}} \rceil$ である．2 本の並列の線のうち，上の線は $R_{\mathrm{kout}} + 8$ bit，下の線は $b - (R_{\mathrm{kout}} + 8)$ bit である．
7. 暗号文を $C = C_1 \| \cdots \| C_m$，タグを $T = \mathsf{msb}_t(Z_1 \| \cdots \| Z_d)$ とする．

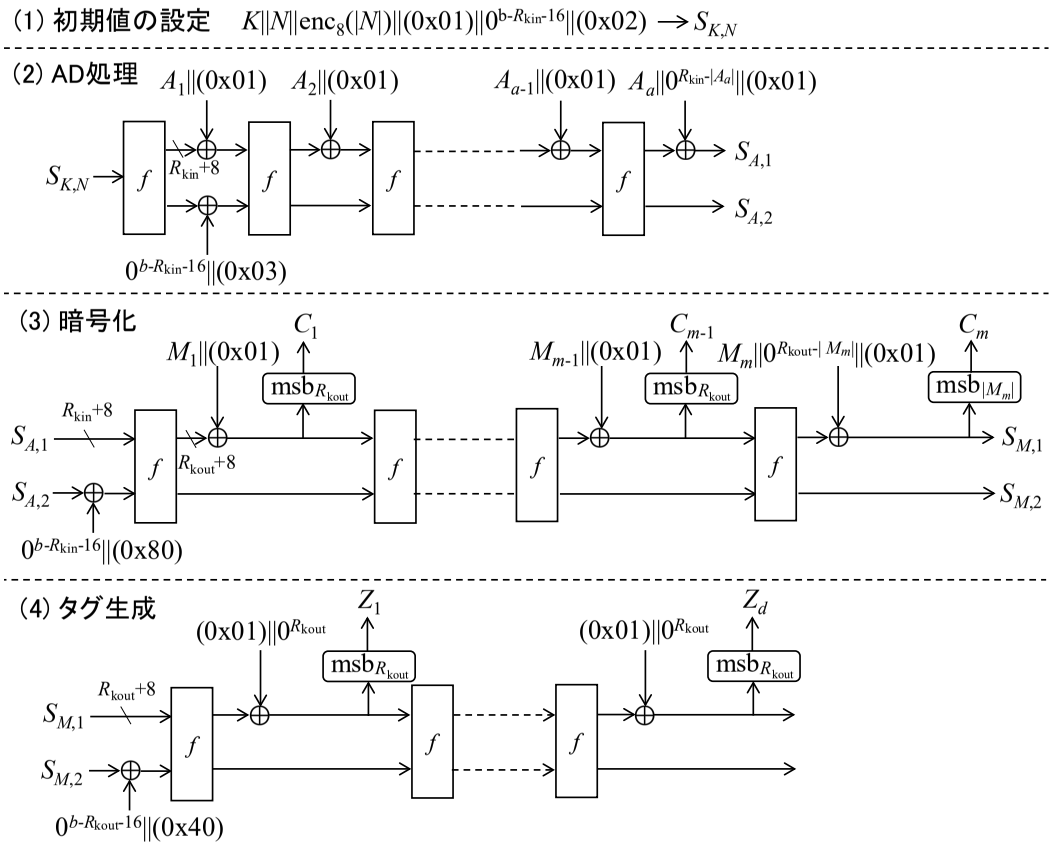


図 4: Xoodyak の認証暗号利用モードの暗号化関数 Xoodyak.$\mathcal{E}$ の手続き

5.2.2 Xoodyak の認証暗号利用モードの復号関数 Xoodyak.$\mathcal{D}$

Xoodyak の認証暗号利用モードの復号関数 Xoodyak.$\mathcal{D}$ の入出力は以下の通りである．

入力

- 鍵 $K \in \{0,1\}^k$
- Associated Data $A \in \{0,1\}^*$
- 暗号文 $C \in \{0,1\}^{|M|}$
- タグ $T' \in \{0,1\}^t$

出力

- 暗号文 $M \in \{0,1\}^{|C|}$，または，タグの改ざんを示す記号 $\perp \notin \{0,1\}^{|C|}$

Xoodyak の認証暗号利用モードの復号関数の手続きは以下の通りである．ここで，Xoodyak.$\mathcal{D}$ の出力値を Xoodyak.$\mathcal{D}_K(N, A, C, T')$ と書くことにする．

1. 図 5(1) を実行し，鍵 K，ナンス N から初期値 $S_{K,N}$ を生成する．
2. Associated Data A を R_{kin} bit のデータブロック $A_1, A_2, \ldots, A_a$ に分割する．ここで，$|A_1| = \cdots = |A_{a-1}| = R_{\mathrm{kin}}$，$|A_a| \leq R_{\mathrm{kin}}$ であり，$A = A_1 \| A_2 \| \cdots \| A_a$，$a = \lceil |A|/R_{\mathrm{kin}} \rceil$ である．
3. 図 5(2) を実行する．ここでは，Associated Data ブロック $A_1, A_2, \ldots, A_a$ を処理し，$S_{A,1}$ と $S_{A,2}$ を生成する．$|S_{A,1}| = R_{\mathrm{kin}} + 8$ と $|S_{A,2}| = b - (R_{\mathrm{kin}} + 8)$ であり，1 回目の f を計算して以降の 2 本の並列の線のうち，上の線は $R_{\mathrm{kin}} + 8$ bit，下の線は $b - (R_{\mathrm{kin}} + 8)$ bit である．
4. 暗号文 C を R_{kout} bit のデータブロック $C_1, C_2, \ldots, C_c$ に分割する．ここで，$|C_1| = \cdots = |C_{m-1}| = R_{\mathrm{kout}}$，$|C_m| \leq R_{\mathrm{kout}}$ であり，$C = C_1 \| C_2 \| \cdots \| C_c$，$c = \lceil |C|/R_{\mathrm{kout}} \rceil$ である．
5. 図 5(3) を実行する．ここでは，暗号文ブロック $C_1, \cdots, C_c$ を復号し，平文ブロック $M_1, \cdots, M_c$，そして，$S_{C,1}$，$S_{C,2}$ を生成する．各 $i \in [c]$ に対し，$|M_i| = |C_i|$，$|S_{C,1}| = R_{\mathrm{kout}} + 8$，$|S_{C,2}| = b - (R_{\mathrm{kout}} + 8)$ であり，1 回目の f を計算して以降の 2 本の並列の線のうち，上の線は $R_{\mathrm{kout}} + 8$ bit，下の線は $b - (R_{\mathrm{kout}} + 8)$ bit である．
6. 図 5(4) を実行し，出力ブロック $Z_1, \ldots, Z_d$ を生成する．$d = \lceil t/R_{\mathrm{kout}} \rceil$ である．2 本の並列の線のうち，上の線は $R_{\mathrm{kout}} + 8$ bit，下の線は $b - (R_{\mathrm{kout}} + 8)$ bit である．
7. 平文を $M = M_1 \| \cdots \| M_c$，タグを $T = \mathsf{msb}_t(Z_1 \| \cdots \| Z_d)$ とする．

8. $T = T'$ ならば Xoodyak.$\mathcal{D}_K(N, A, C, T') = M$ とする．

9. $T \neq T'$ ならば Xoodyak.$\mathcal{D}_K(N, A, C, T') = \mathsf{reject}$ とする．

5.3 Xoodoo[12] の仕様

Xoodoo[12] は 384bit の置換であり，ラウンド関数を 12 回繰り返す構造を持つ．R_i を i ラウンド目のラウンド関数とする．$i \in [12]$ で，各ラウンド関数 R_i は 384bit の置換である．

5.3.1 ステートの状態

Xoodoo[12] では，384bit のステート S を各ラウンド関数 R_i の内で 3 つの 128bit の値 S_0, S_1, S_2 に分割する．$S = S_0 \| S_1 \| S_2$ である．各 128bit の値を plane と呼び，このインデックスを $y \in (2]$ で表現する．さらに，各 planeS_y を 4 つの 32bit の値 $S_{0,y}, S_{1,y}, S_{2,y}, S_{3,y}$ に分割する．$S_y = S_{0,y}, \| S_{1,y} \| S_{2,y} \| S_{3,y}$ である．各 32bit の値を lane と呼び，このインデックスを $x \in (4]$ で表現する．そして，各 32bit の lane$S_{x,y}$ の z bit 目を $S_{x,y,z}$ とする．$z \in (31]$ であり，$S_{x,y} = S_{x,y,0} \| S_{x,y,1} \| \cdots \| S_{x,y,31}$ である．

5.4 Xoodoo[12] で用いられる演算

Xoodoo[12] の各ラウンド関数 R_i では以下の演算が用いられる．

- $S_y \lll (t, v)$: 各 $(x, z) \in (3] \times (31]$ に対し，$S'_{x+t,y,z+v} \leftarrow S_{x,y,z}$ とし，その後，各 $(x, z) \in (3] \times (31]$ に対し，$S_{x,y,z} \leftarrow S'_{x,y,z}$ とする．
- $\overline{S_y}$: S_y のビット単位の補数．S_y の各ビット 0 は 1，1 は 0 とする．
- $S_y + S_{y'}$: S_y と $S_{y'}$ のビット単位で XOR 演算をした値.
- $S_y \cdot S_{y'}$: S_y と $S_{y'}$ のビット単位で AND 演算をした値.

5.4.1 Xoodoo[12] の入出力と手続き

Xoodoo[12] の入出力は以下の通りである．

入力

- $X \in \{0, 1\}^{384}$

出力

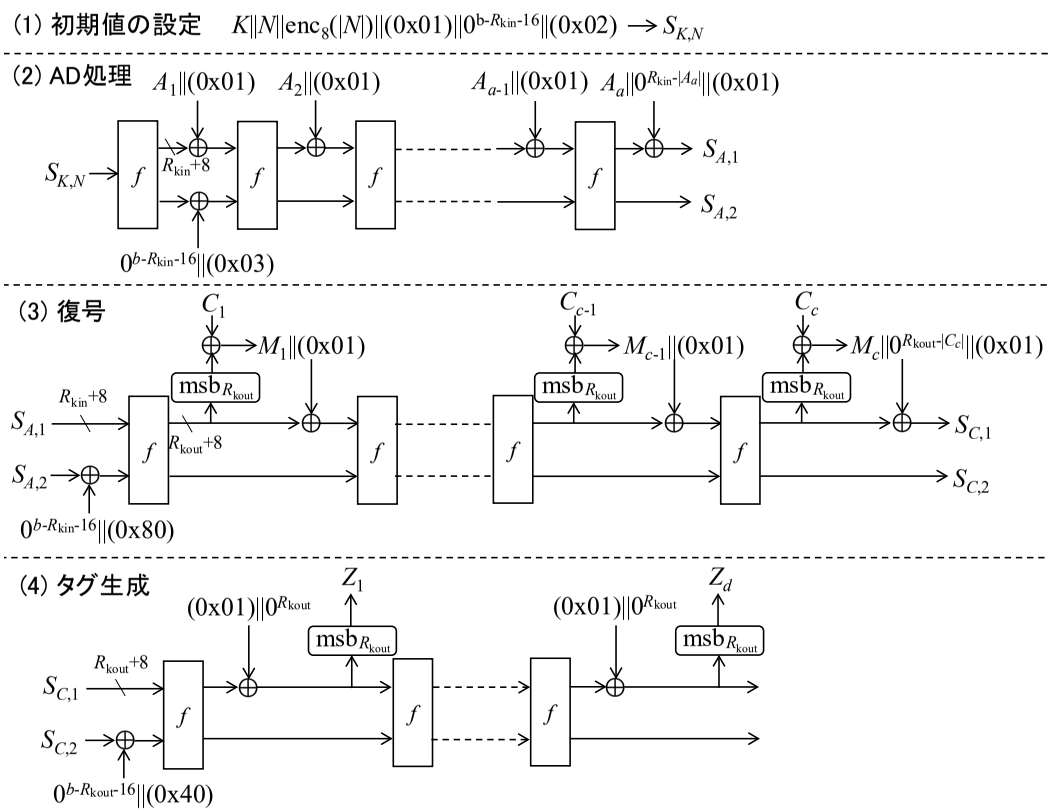


図 5: Xoodyak の認証暗号利用モードの復号関数 Xoodyak.$\mathcal{D}$ の手続き

- $Y \in \{0,1\}^{384}$

Xoodoo[12] の手続きは以下の通りである．

1. ステート S の初期値として X を代入する．
2. $i = 1, \ldots, 12$ に対して，$S = R_i(S)$ を計算する．
3. Xoodoo[12] の出力を $Y = S$ とする．

次に，384bit のステート S に対して，ラウンド関数 R_i の仕様を以下に示す．なお，各固定値 C_i は文献 [11] を参照されたい．

1. $P \leftarrow S_0 + S_1 + S_2$
2. $E \leftarrow P \lll (1,5) + P \lll (1,14)$
3. $y = 0,1,2$ に対して，$S_y \leftarrow S_y + E$ とする．
4. $S_1 \leftarrow S_1 \lll (1,0)$ とする．
5. $S_2 \leftarrow S \lll (0,11)$ とする．
6. $S_0 \leftarrow S_0 + C_i$ とする．
7. $B_0 \leftarrow \overline{S_1} \cdot S_2$ とする．
8. $B_1 \leftarrow \overline{S_2} \cdot S_0$ とする．
9. $B_2 \leftarrow \overline{S_0} \cdot S_1$ とする．
10. $y = 0,1,2$ に対して，$S_y \leftarrow S_y + B_y$ とする．
11. $S_1 \leftarrow S_1 \lll (0,1)$ とする．
12. $S_2 \leftarrow S_2 \lll (2,8)$ とする．

6 Xoodyak の安全性に関する文献調査の結果

Xoodyak のハッシュ関数と認証暗号の安全性評価は，Xoodoo[12] をブラックボックスとする場合の評価と，Xoodoo[12] の構造を入れた評価がある．以下，Xoodoo[12] の安全性，Xoodyak のハッシュ関数の Xoodoo[12] をブラックボックスとする場合の安全性，Xoodoo[12] の構造を入れる場合の安全性，Xoodyak の認証暗号の Xoodoo[12] をブラックボックスとする場合の安全性，Xoodoo[12] の構造を入れる場合の安全性に関する文献調査の結果を報告する．

6.1 Xoodoo[12] の安全性

文献 [16] は，Xoodoo[12] に対して，オフライン計算量が 2^{33} の Zero-sum Distinguisher を構成している．ただし，文献 [11] で設計者が述べているように，この攻撃が Xoodyak の安全性に直接影響を及ぼすものではないことに注意されたい．

6.2 Xoodyak のハッシュ関数の安全性

Xoodyak の安全性評価に関して，Xoodyak のプリミティブ Xoodoo[12] は単体で安全では無く，Xoodoo[12] がブラックボックスの場合の Xoodyak のハッシュ関数の安全性評価と，Xoodoo[12] の構造を含めた Xoodyak のハッシュ関数の安全性評価が必要である．以下，Xoodoo[12] と Xoodyak のハッシュ関数の Xoodoo[12] をブラックボックスとする場合とそうでない場合の安全性に関する文献調査の結果を報告する．

6.2.1 Xoodoo[12] がブラックボックスの場合の安全性

Xoodyak のハッシュ関数は，利用モードとして Sponge 構造 [6] を採用している．ここで，$r = R_{\mathrm{hash}}$，$c = b - r$ とする．Sponge 構造は，置換ベースのハッシュ関数利用モードであり，置換 f がランダム置換の場合，Indifferentiablity の安全性を持つことが証明されている．具体的には，以下の定理が文献 [6] で示されている．

Theorem 2 (Sponge の Indifferentiability)**.** 全クエリーのランダム置換の呼び出し回数が σ 回の *Indifferentiability* の攻撃者 A に対し，以下が成り立つ．

$$\mathsf{Adv}^{\mathsf{indiff}}_{\mathsf{Sponge}}(\mathsf{A}) \leq 1 - \prod_{i \in (\sigma - 1]} \frac{1 - \frac{i+1}{2^c}}{1 - \frac{i}{2^b}} \ . \blacksquare$$

上記のバウンドはおよそ $\frac{(1-2^{-r})\sigma^2 + (1+2^{-r})\sigma}{2^{c+1}}$ であり，攻撃者の計算量が $\sigma = 2^{c/2}$ の場合にバウンドがコンスタントとなるため，Sponge 構造は，Indifferentiability に関して，$c/2$ bit 安全である．

Indifferentiability 以外の安全性に関して，Indifferentiability 安全なハッシュ関数は，$c/2$ とランダムオラクルの安全性レベルの最小の値が安全性レベルとなる．よって，Collision Resistance に関して $\min\{\ell/2, c/2\}$ bit，Second Preimage Resistance に関して $\min\{\ell, c/2\}$ bit，Preimage Resistance に関して $\min\{\ell, c/2\}$ bit，m-target preimage resistance に関して $\min\{\ell - \log_2 m, c/2\}$ bit の安全性を持つ．$R_{\mathrm{hash}} = 128$，$b = 384$，$\ell = 256$ なので，Xoodyak のハッシュ関数は，Xoodoo[12] をブラックボックスとする場合，設計者が主張する安全性を持つ．

6.2.2 Xoodoo[12] の構造を入れる場合の安全性

Xoodoo[12] の構造を考慮に入れる場合の安全性は，今のところ解析論文は無く，設計者が主張する安全性を満たすと考えられる．

6.3 Xoodyak の認証暗号の安全性

Xoodyak の安全性評価に関して，Xoodyak のプリミティブ Xoodoo[12] は単体で安全では無く，Xoodoo[12] がブラックボックスの場合の Xoodyak の認証暗号の安全性評価と，Xoodoo[12] の構造を含めた Xoodyak の認証暗号の安全性評価が必要である．以下，Xoodoo[12] と Xoodyak の認証暗号の Xoodoo[12] をブラックボックスとする場合とそうでない場合の安全性に関する文献調査の結果を報告する．

6.3.1 設計者が主張する Xoodoo[12] がブラックボックスの場合の安全性

Xoodyak の安全性に関して，設計者は AE 安全性に関して以下のバウンドが成り立つと主張している．以下の Claim は，文献 [11] の Claim 2 から導出されたものである．

Claim 3 (Xoodyak の AE 安全性)**.** Xoodyak の置換 f をランダム関数とする．この時，ナンスリスペクトの Xoodyak の AE 安全性を破る任意の攻撃者 A に対し，以下の不等式が成り立つ．ここで，A のオンラインクエリの計算量を σ，オフラインクエリ回数を p とする．

$$\mathsf{Adv}^{\mathsf{ae}}_{\mathrm{Xoodyak}}(\mathsf{A}) \leq \frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma^2}{2^{192+k}} \ . \ \blacksquare$$

Claim 3 から，Xoodyak の認証暗号の秘匿性に関して以下のバウンドが得られる．

Claim 4 (Xoodyak の秘匿性)**.** Xoodyak の置換 f をランダム関数とする．この時，ナンスリスペクトの Xoodyak の秘匿性を破る任意の攻撃者 A に対し，以下の不等式が成り立つ．ここで，A のオンラインクエリの計算量を σ_e，オフラインクエリ回数を p とする．

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathrm{Xoodyak}}(\mathsf{A}) \leq \frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma_e^2}{2^{192+k}} \ . \ \blacksquare$$

また，Claim 3 から，Xoodyak の認証暗号の偽造不可能性に関して以下のバウンドが得られる．

Claim 5 (Xoodyak の偽造不可能性)**.** Xoodyak の置換 f をランダム関数とする．この時，ナンスリスペクトの Xoodyak の AE 安全性を破る任意の攻撃

者 A に対し，以下の不等式が成り立つ．ここで，A のオンラインクエリの計算量を σ，オフラインクエリ回数を p とする．

$$\mathsf{Adv}^{\mathsf{auth}}_{\mathrm{Xoodyak}}(\mathsf{A}) \leq \frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma^2}{2^{192+k}} \ . \ \blacksquare$$

Xoodyak は $k = 128$ なので，設計者が主張する，秘匿性と偽造不可能性について，オンライン計算量では 160bit，オフライン計算では 128bit の安全性が得られる．

6.3.2 Xoodoo[12] がブラックボックスの場合の安全性に関する考察

文献 [11] の Claim 2 では，Xoodyak の一般化アルゴリズム Cyclist がランダム関数と識別不可能性を示しており，このバウンドを Xoodyak に適用すると，Xoodyak の AE 安全性のバウンドは以下のようになる．

$$\frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma^2}{2^{192+k}} + \frac{(L+\Omega)p + \binom{L+\Omega+1}{2}}{2^{192}} \ . \tag{1}$$

ここで，L は攻撃者が選んだデータブロックが Xoodyak 内部で呼ばれる置換の R_{kout} bit 部分に直接入力される回数であり，Ω は Xoodyak への各クエリの置換の入力を先頭から結合した b bit ブロックの系列を見た場合，先頭から 1 ブロック以上同じブロックが現れる系列のペアの個数である．例えば，ナンスと鍵が同じで，次のデータブロックが違うクエリは 1 ペアと数える．

Xoodyak の設計者は，式 (1) のバウンドで，$L = 0, \Omega = 0$ と設定して，Claim 3 のバウンドを導出している．Xoodyak の設計者は，復号オラクルが Unverified Plaintext をタグの検証に合格しない時に出力しないならば，$L = 0, \Omega = 0$ であると主張している．ナンスリスペクトのセッティングで，暗号化オラクルへのクエリのみを考えるならば，$L = 0, \Omega = 0$ は正しいが，暗号化オラクルと復号オラクルへのクエリを両方考える場合は正しくない．攻撃者は復号オラクルへのクエリはナンスを含め自由に選択することができるため，ある暗号化クエリのナンスと同じナンスを全ての復号クエリで投げると $\Omega \geq q_d$ が成立する．q_d は復号オラクルへのクエリ回数である．また，復号関数では，暗号文ブロックが直接置換の入力値となることから，$L \geq \sigma_d$ が成立する．σ_d は復号クエリで呼び出される置換の呼び出し回数である．$\Omega = q_d, L = \sigma_d$ の場合，式 (1) は以下のようになる．

$$\frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma^2}{2^{192+k}} + \frac{(q_d+\sigma_d)p + 0.5(q_d+\sigma_d+1)^2}{2^{192}} \ .$$

このバウンドから，以下の秘匿性のバウンドが得られる．

$$\frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma_e^2}{2^{192+k}} \ .$$

そして，以下の偽造不可能性のバウンドが得られる．

$$\frac{p}{2^k} + \frac{p}{2^{184}} + \frac{\sigma^2}{2^{192+k}} + \frac{(q_d+\sigma_d)p + 0.5(q_d+\sigma_d+1)^2}{2^{192}} \ .$$

上記の秘匿性のバウンドは，設計者が主張する秘匿性の安全性レベルと矛盾しない．しかし，上記の偽造不可能性のバウンドは，オフライン計算で 128 bit 安全の時，オンライン計算で 64 bit の安全性までしか保証できないため，設計者が主張する偽造不可能性のオンライン計算に関する安全性レベル 160 bit は上記の考察と矛盾する．

今のところ，160 bit の偽造不可能性を破る攻撃は存在しないが，安全性証明と設計者の主張に差があることに注意が必要である．

6.3.3 Xoodoo[12] の構造を入れる場合の安全性について

Xoodoo[12] の構造を入れる場合の Xoodyak の認証暗号に対する解析論文はいくつか発表されているが，7 ラウンド以上の攻撃は存在しない．以下，調査結果を記載する．

Conditional Cube 攻撃

文献 [24] で，Conditional Cube 攻撃で，Xoodoo[12] のラウンド数を 6 ラウンドに削減した Xoodyak の認証暗号の鍵復元に成功している．鍵復元に必要な時間は，2^{44} である．

差分線形攻撃

文献 [13] で，4 ラウンド Xoodoo[12] の差分線形の Distinguisher を構成し，この Distinguisher を用いて 4 ラウンド Xoodoo[12] を用いる認証暗号の鍵復元に成功している．鍵復元に必要なオフライン計算量は，$2^{23.34}$ である．5 ラウンド Xoodoo[12] の差分線形の Distinguisher を構成し，この Distinguisher を用い 5 ラウンド Xoodoo[12] を用いる認証暗号の鍵復元に成功している．鍵復元に必要なオフラインの計算量は $2^{22.04}$ である．

7 結論

本報告書では，GIFT-COFB [2, 3, 1] と Xoodyak [11, 10] の仕様を記載し，これらの方式の安全性に関する文献調査を報告した．

GIFT-COFB の安全性に関して，GIFT-COFB の利用モードである COFB の安全性は文献 [8, 4] で証明されており，調査時点で間違いは無いと考えられる．また，GIFT-COFB のプリミティブである GIFT-128 も調査時点で 40

ラウンド中 27 ラウンドまでしか攻撃されておらず，十分に安全性のマージンがあるため，GIFT-128 は安全であると考えられる．以上より，GIFT-COFB は設計者が主張する安全性を満たすと考えられる．なお，文献 [9, 3] に COFB の安全性のバウンドが与えられているが，文献 [15, 14] でこれらのバウンドが間違いであることが示されている．

Xoodyak のハッシュ関数の安全性に関して，Xoodoo[12] をブラックボックスとする場合，Sponge 構造の安全性証明 [6] からその安全性は保証され，Xoodoo[12] の構造を入れる場合，調査時点で解析結果は報告されていない．以上より，Xoodyak のハッシュ関数の安全性は設計者が主張する安全性を満たすと考えられる．

Xoodyak の認証暗号の安全性に関して，Xoodoo[12] をブラックボックスとする場合，設計者は Duplex 構造の安全性証明からその安全性を主張しているが，Duplex 構造の安全性バウンドと設計者が主張する安全性には差がある．なお，調査時点では，設計者が主張する安全性を破る攻撃法は見つかっていない．Xoodoo[12] の構造を入れる場合，調査時点で 12 ラウンド中 6 ラウンドまでしか攻撃されておらず，十分に安全性のマージンがあるため，安全であると考えられる．以上より，Xoodyak の認証暗号は設計者が主張する安全性を満たすと考えられるが，Xoodoo[12] をブラックボックスとする場合の安全性の主張に矛盾点があるため注意が必要である．

参考文献

[1] Subhadeep Banik, Avik Chakraborti, Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB Authenticated Encryption .

[2] Subhadeep Banik, Avik Chakraborti, Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB v1.2.

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

[4] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB. *IACR Cryptol. ePrint Arch.*, page 738, 2020.

[5] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[6] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[7] 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, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

[8] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 277–298. Springer, 2017.

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

[10] Joan Daemen, Seth Hoffert, Silvia Mella, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak.

[11] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Silvia Mella. Xoodyak, a lightweight cryptographic scheme. *A Submission to the NIST Lightweight Cryptography Standardization Process (2021),*.

[12] 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, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II*, volume 10625 of *Lecture Notes in Computer Science*, pages 606–637. Springer, 2017.

[13] Orr Dunkelman and Ariel Weizman. Differential-linear cryptanalysis on xoodyak. In *NIST LWC Workshop 2022*, 2022.

[14] Akiko Inoue, Tetsu Iwata, and Kazuhiko Minematsu. Analyzing the provable security bounds of GIFT-COFB and photon-beetle. In Giuseppe Ateniese and Daniele Venturi, editors, *Applied Cryptography and Network Security - 20th International Conference, ACNS 2022, Rome, Italy, June 20-23, 2022, Proceedings*, volume 13269 of *Lecture Notes in Computer Science*, pages 67–84. Springer, 2022.

[15] Mustafa Khairallah. Security of COFB against chosen ciphertext attacks. *IACR Trans. Symmetric Cryptol.*, 2022(1):138–157, 2022.

[16] Fukang Liu, Takanori Isobe, Willi Meier, and Zhonghao Yang. Algebraic attacks on round-reduced keccak/xoodoo. *IACR Cryptol. ePrint Arch.*, page 346, 2020.

[17] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[18] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 257–274. Springer, 2014.

[19] NIST. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process, 2018.

[20] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

[21] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[22] Ling Sun, Wei Wang, and Meiqin Wang. Linear cryptanalyses of three aeads with GIFT-128 as underlying primitives. *IACR Trans. Symmetric Cryptol.*, 2021(2):199–221, 2021.

[23] Ling Sun, Wei Wang, and Meiqin Wang. Addendum to linear cryptanalyses of three aeads with GIFT-128 as underlying primitives. *IACR Cryptol. ePrint Arch.*, page 151, 2022.

[24] 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.

[25] Rui Zong, Xiaoyang Dong, Huaifeng Chen, Yiyuan Luo, Si Wang, and Zheng Li. Towards key-recovery-attack friendly distinguishers: Application to GIFT-128. *IACR Trans. Symmetric Cryptol.*, 2021(1):156–184, 2021.