# Lightweight Authenticated Encryption Mode of Operation for Tweakable Block Ciphers

Yusuke Naito and Takeshi Sugawara

Mitsubishi Electric Corporation, Japan
The University of Electro-Communications, Japan

**Abstract.** Using a small block length is a common strategy in designing lightweight block cipher. So far, many 64-bit primitives have been proposed. However, if we use such a 64-bit primitive for an authenticated encryption with birthday-bound security, it has only 32-bit plaintext complexity which is subject to a practical attack. To take advantage of a short block length without losing security, we propose a lightweight AEAD mode FBAE that achieves beyond-birthday-bound security. For the purpose, we extend the idea of iCOFB, originally defined with a tweakable random function, with tweakable block cipher. More specifically, we fix the tweak length which was variable in iCOFB, and further generalize the feedback function. Moreover, we improve its security bound. We evaluate the concrete hardware performances of FBAE. FBAE benefits from the small block length and shows the particularly good performances in threshold implementation.

**Keywords:** Authenticated encryption, beyond-birthday-bound security, tweakable block-cipher, lightweight, threshold implementation.

## 1 Introduction

Driven by a demand for secure connectivity in resource-constrained embedded devices, lightweight cryptography has been actively studied in the last decade. Consequently, a number of lightweight block ciphers have been proposed [2, 5, 6, 9, 12, 34] including PRESENT [8, 36] and CLEFIA [35] standardized in ISO/IEC 29192-2.

A common strategy for designing a lightweight block cipher is to use a small block length. For example, PRESENT [8] and PRINCE [9] support 64-bit block length only. Many more algorithms such as GIFT [4] and SKINNY [6] provide 64-bit options. The small block length contributes to a smaller memory footprint and a shorter round number that is crucial for a lightweight implementation.

Resource-constrained devices are frequently used in a hostile environment in which side-channel attack (SCA) [18] should be considered. Designers face an even more challenging task of realizing an SCA-resistant implementation with a limited resource. Researchers have tackled this problem and proposed many lightweight and SCA resistant implementations [28, 32, 24, 6, 13] including the ones protected by threshold implementation (TI) [29]. The advantage of a block cipher with a small block length (i.e., a small state size) becomes even larger with TI in which a shared representation of the state multiplies the memory requirement.

In order to leverage the benefit of lightweight block cipher for realizing both confidentiality and integrity, lightweight modes of operation for authenticated encryption with associated data (AEAD) have been actively studied in the last few years promoted by the CAESAR competition and the NIST's move toward standardizing lightweight cryptography [30]. So far, lightweight AEAD modes such as COFB [10] and SAEB [25] have been proposed. However, the short block length of lightweight cryptography can be a problem for security. The lightweight AEAD modes

**Table 1.** The lightweight criteria [25] and AEAD modes. The "No extra state" column shows the number of extra bits if the criterion is not satisfied.

| AEAD | Primitive | Security | Lightweight criteria | | | | Ref. |
|---|---|---|---|---|---|---|---|
| | | | No extra state | Inv. free | XOR only | Online | |
| COFB | Block cipher | $O(2^{b/2})$ | $b/2$ | ✓ | — | ✓ | [10] |
| SAEB | Block cipher | $O(2^{b/2})$ | ✓ | ✓ | ✓ | ✓ | [25] |
| ΘCB3 | TBC | $O(2^b)$ | $2b$ | — | — | ✓ | [19] |
| FBAE | TBC | $O(2^b)$ | ✓ | ✓ | ✓ | ✓ | **Ours** |

have security up to the so-called birthday bound. More specifically, the security is ensured up to $O(2^{b/2})$ block-cipher calls when instantiated with a $b$-bit block cipher. With a 64-bit block cipher, the security is ensured up to $2^{32}$ block-cipher calls only. It is subject to a practical attack as demonstrated by the Sweet32 attack [7].

The use of an AEAD mode with beyond-birthday-bound (BBB) security is a solution for avoiding the birthday problem. There are block-cipher-based AEAD modes with BBB security including CHM [14], CIP [15], and AEAD modes with CLRW2 [21] or $r$-CLRW [20]. However, they are costly compared with the lightweight AEAD modes, since two or more independent universal hash functions are required. Another solution is to construct a (dedicated) TBC-based AEAD mode. The TBC-based AEAD modes, including ΘCB3 [19], $\mathbb{OTR}$ [22], SCT [31] and ZAE [16], realize better efficiency and security. Especially, ΘCB3 has the smallest state in the category of the BBB-secure AEAD modes.

### 1.1 Motivation, Approach, and Problems

Our motivation is to design a lightweight BBB-secure AEAD mode thereby taking advantage of a short block length without losing security. For being lightweight, we use the four criteria for lightweight AEAD [25] which is used in designing the block-cipher-based lightweight AEAD mode SAEB as shown in Table 1:

- **No extra state**: The AEAD mode uses no additional memory in addition to the ones used within the (tweakable) block cipher.
- **Inverse free**: The AEAD mode uses no decryption call of the (tweakable) block cipher.
- **XOR only**: The AEAD mode needs XOR only in addition to the (tweakable) block cipher.
- **Online**: The AEAD mode scans the incoming message only once.

Using a (dedicated) TBC is a promising approach for designing a lightweight and BBB-secure AEAD mode; however, none of the previous TBC-based AEAD modes, including ΘCB3, satisfy all the lightweight criteria (see Table 1).

Our approach is to design a (dedicated) TBC-based AEAD mode by extending the idea of iCOFB [10]. iCOFB shown in Figure 1 is a generalization of COFB by tweakable random function (TRF) having $b$-bit outputs, denoted by $R$. In iCOFB, a TRF is called for each message/ciphertext block. Feedback functions $\rho/\rho'$ are used to map a pair of a TRF output $Y_i$ and a plaintext/ciphertext block $M_i/C_i$ to the next TRF input $X_{i+1}$ and a ciphertext/plaintext block $C_i/M_i$. More specifically, the following linear functions are considered, which is expressed

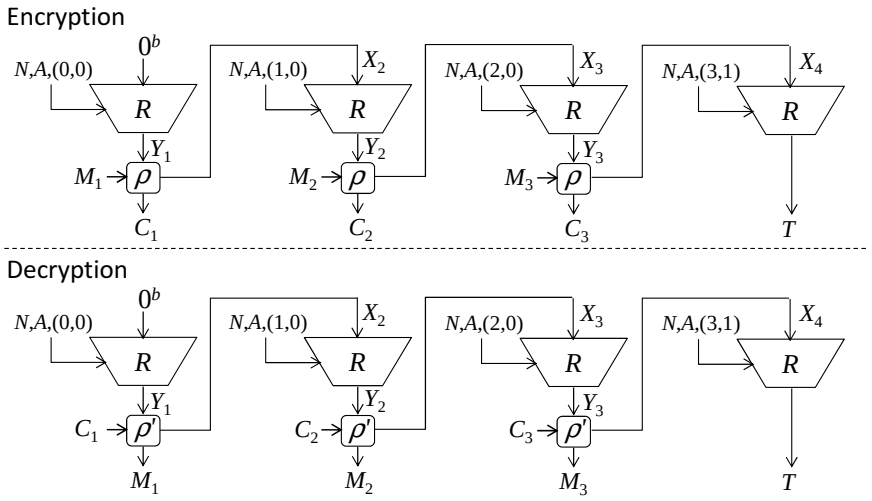


**Fig. 1.** iCOFB. $M_1, M_2, M_3$ are $b$-bit plaintext blocks, $C_1, C_2, C_3$ are $b$-bit ciphertext blocks, and $T$ is a $b$-bit tag.

by a $2b \times 2b$ binary matrix.

$$\rho(Y_i, M_i) = \begin{pmatrix} X_{i+1} \\ C_i \end{pmatrix} = \begin{pmatrix} E_{1,1}\ E_{1,2} \\ E_{2,1}\ E_{2,2} \end{pmatrix} \begin{pmatrix} Y_i \\ M_i \end{pmatrix} \ ,$$
$$\rho'(Y_i, C_i) = \begin{pmatrix} X_{i+1} \\ M_i \end{pmatrix} = \begin{pmatrix} D_{1,1}\ D_{1,2} \\ D_{2,1}\ D_{2,2} \end{pmatrix} \begin{pmatrix} Y_i \\ C_i \end{pmatrix} \ .$$

After consuming all the message blocks, a TRF is called once again to generate a tag $T$. It was proven that iCOFB has $O(2^b)$ security with $(\rho,\ \rho')$ satisfying a certain criterion (see Subsection 3.1).

As shown in Figure 1, iCOFB needs no extra state in addition to the ones within the underlying TRF. Besides, iCOFB takes message/ciphertext blocks online and does not need an inverse of TRF. Moreover, the linear functions $\rho$ and $\rho'$ can be realized with XOR only. Therefore, iCOFB satisfies all the requirements regarding TRF-based AEAD.

There are two problems in designing a TBC-based AEAD mode from the iCOFB's idea. First, since associated data (AD) is a part of a tweak, the underlying TRF should accept an arbitrary-length tweak. On the contrary, lightweight TBCs such as SKINNY accepts a fixed-length tweak only. Using the XT tweak extension [23] is a possible solution, but it requires a universal hash function accepting an arbitrary-length input that can be costly in implementation. Second, the security bound of iCOFB is $O(\ell_{\mathsf{max}} q/2^b)$ which depends on the maximum message block length $\ell_{\mathsf{max}}$ and the number of queries $q$ (the sum of the numbers of encryption queries and forgery attempts). It is degraded compared with that of ΘCB3, $O(q_{\mathcal{D}}/2^b)$, wherein $q_{\mathcal{D}}$ is the number of forgery attempts. Large $\ell_{\mathsf{max}}$ and/or $q$ cause a short key life: an additional cost for rekeying or a shorter product lifetime.

### 1.2 Contribution

We design a (fixed tweak-length) TBC-based AEAD mode called FBAE that solves the above two problems and satisfies all the lightweight criteria as shown in Table 1. Moreover, we general-

ize the feedback functions that cover a broader class of feedback functions including non-linear ones.

We address the first problem by designing a new AD processing part. We introduce a (possibly non-linear) feedback function $\delta^{(a)}$ that maps an AD block $A_i$ and an TBC output block $W_i$ to the next TBC input $V_{i+1}$. A given AD is processed block by block by using a fixed-tweak TBC and the feedback function $\delta^{(a)}$.

To address the second problem, we generalize the feedback functions $\rho$ and $\rho'$ to the pairs of functions $(\gamma^{(e)}, \delta^{(e)})$ and $(\gamma^{(d)}, \delta^{(d)})$ given by

$$\begin{aligned} \delta^{(e)} &: (Y_i, M_i) \mapsto X_{i+1}, & \gamma^{(e)} &: (Y_i, M_i) \mapsto C_i, \\ \delta^{(d)} &: (Y_i, C_i) \mapsto X_{i+1}, & \gamma^{(d)} &: (Y_i, C_i) \mapsto M_i. \end{aligned}$$

We show conditions on the generalized feedback functions (given in Subsections 3.2 and 3.3) under which FBAE satisfy the security bound of $O(q_{\mathcal{D}}/2^b)$ — the same level of security as ΘCB3. The set of generalized feedback function satisfying the condition is a superset of $(\rho, \rho')$ in iCOFB, and thus involves a broader class of functions.

The benefit of the proposed TBC-based AEAD mode is evaluated through concrete hardware implementations. In the implementation, we use a particularly efficient set of feedback functions:

$$\begin{aligned} \delta^{(a)}(W_i, A_i) &= W_i \oplus A_i, \\ \delta^{(e)}(Y_i, M_i) &= M_i, & \gamma^{(e)}(Y_i, M_i) &= Y_i \oplus M_i, \\ \delta^{(d)}(Y_i, C_i) &= Y_i \oplus C_i = M_i, & \gamma^{(d)}(Y_i, C_i) &= Y_i \oplus C_i. \end{aligned}$$

We refer the specialization as the plaintext feedback mode (PFB) because the TBC input is always $M_i$. We remark that the encryption of PFB is parallelizable, unlike the existing lightweight AEAD modes COFB and SAEB[1]. The feature is desirable for communication between entities with asymmetric resources, e.g., a central server sends encrypted commands to many resource-constrained nodes.

In the implementations, PFB is instantiated with the lightweight TBC SKINNY-64-192. Its performance is compared with the state-of-the-art block-cipher-based alternative with the same level of security: SAEB instantiated with GIFT-128-128. For each of the AEADs, we evaluate the performances with and without TI. We show that PFB benefit from the small block length and shows the particularly good performance in implementations with the SCA countermeasure: it has the smallest circuit area compared with the SAEB implementation and the conventional implementations of Ascon [11]) and Ketje [1].

### 1.3 Organization

This paper is organized as follows. In Section 2, we briefly review TBC and AEAD. Then, we describe the design principle and definition of FBAE in Section 3, followed by its security result is Section 4. We show hardware implementations and their performance comparison in Section 5.

[1] The decryption of PFB is not parallelizable, whereas both the encryption and decryption of ΘCB3 is parallelizable. However, as shown in Table 1, ΘCB3 does not satisfy three out of the four conditions. Regarding $\rho$ and $\rho'$, COFB uses the other feedback functions called combined feedback, which does not offer the parallelizability. SAEB has the iterated structure of a block cipher, thus does not has the parallelizability.

## 2 Preliminaries

**Notation.** Let $\lambda$ be an empty string and $\{0,1\}^*$ the set of all bit strings. For an integer $i \geq 0$, let $\{0,1\}^i$ be the set of all $i$-bit strings, $\{0,1\}^0 := \{\lambda\}$, $\left(\{0,1\}^i\right)^*$ the set of all bit strings whose lengths are multiples of $i$, and $\{0,1\}^{\leq i} := \{0,1\}^1 \cup \{0,1\}^2 \cup \cdots \cup \{0,1\}^i$ the set of all bit strings of length at most $i$. Let $0^i$ resp. $1^i$ be the bit string of $i$-bit zeros resp. ones. For an integer $i \geq 1$, let $[i] := \{1, 2, \ldots, i\}$ be the set of positive integers equal to or less than $i$, and $(i) := \{0\} \cup [i]$. For a non-empty set $\mathcal{T}$, $T \xleftarrow{\$} \mathcal{T}$ means that an element is chosen uniformly at random from $\mathcal{T}$ and is assigned to $T$. The concatenation of two bit strings $X$ and $Y$ is written as $X\|Y$ or $XY$ when no confusion is possible. For integers $0 \leq i \leq j$ and $X \in \{0,1\}^j$, let $\mathsf{msb}_i(X)$ resp. $\mathsf{lsb}_i(X)$ be the most resp. least significant $i$ bits of $X$. For integers $i$ and $j$ with $0 \leq i < 2^j$, let $\mathsf{str}_j(i)$ be the $j$-bit binary representation of $i$. For an integer $b \geq 0$ and a bit string $X$, we denote the parsing into fixed-length $b$-bit strings as $(X_1, X_2, \ldots, X_\ell) \xleftarrow{b} X$, where $X = X_1\|X_2\|\cdots\|X_\ell$, $|X_i| = b$ for $i \in [\ell - 1]$, and $0 < |X_\ell| \leq b$. For an integer $b > 0$, let $\mathsf{ozp}_b : \left(\{\lambda\} \cup \{0,1\}^{\leq b}\right) \to \{0,1\}^b$ be a one-zero padding function: for a bit string $X \in \{0,1\}^{\leq b}$, $\mathsf{ozp}_b(X) = X$ if $|X| = b$; $\mathsf{ozp}_b(X) = X\|10^{b-1-|X|}$ if $|X| < b$.

**Tweakable Block Cipher.** A tweakable block cipher (TBC) is a set of permutations indexed by a key and a public input called tweak. Let $\mathcal{K}$ be the key spece, $\mathcal{TW}$ the tweak space, and $b$ the input/output-block size. A TBC (encryption) is denoted by $\widetilde{E} : \mathcal{K} \times \mathcal{TW} \times \{0,1\}^b \to \{0,1\}^b$. A TBC having a key $K \in \mathcal{K}$ is denoted by $\widetilde{E}_K$, and $\widetilde{E}_K$ having a tweak $TW \in \mathcal{TW}$ is denoted by $\widetilde{E}_K^{TW}$.

In this paper, a keyed TBC is assumed to be a secure tweakable-pseudo-random permutation, or TPRP for short, which is indistinguishable from a tweakable random permutation (TRP). A tweakable permutation (TP) $\widetilde{P} : \mathcal{TW} \times \{0,1\}^b \to \{0,1\}^b$ is a set of $b$-bit permutations indexed by a tweak in $\mathcal{TW}$. A TP having a tweak $TW \in \mathcal{TW}$ is denoted by $\widetilde{P}^{TW}$. Let $\widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^b)$ be the set of all TPs of block size $b$ and tweak space $\mathcal{TW}$. A TRP is defined as $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^b)$. In the TPRP-security game, an adversary $\mathbf{A}$ has access to either the target keyed TBC $\widetilde{E}_K$ for $K \xleftarrow{\$} \mathcal{K}$ or a TRP $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^b)$. After the interaction, $\mathbf{A}$ returns a decision bit $y \in \{0,1\}$. The output of $\mathbf{A}$ with access to $\mathcal{O}$ is denoted by $\mathbf{A}^{\mathcal{O}}$. For a TBC $\widetilde{E}$, the TPRP-security advantage function of an adversary $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\mathbf{A}) := \Pr\left[K \xleftarrow{\$} \mathcal{K}; \mathbf{A}^{\widetilde{E}_K} = 1\right] - \Pr\left[\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}(\mathcal{TW}, \{0,1\}^b); \mathbf{A}^{\widetilde{P}} = 1\right],$$

where the probabilities are taken over $K, \widetilde{P}$ and $\mathbf{A}$.

The maximum over all adversaries, running in time at most $t$ and making at most $\sigma$ queries, is denoted by

$$\mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma, t) := \max_{\mathbf{A}} \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\mathbf{A}) \ .$$

**Nonce-Based Authenticated Encryption with Associated Data.** A nonce-based authenticated encryption with associated data (nAEAD) scheme based on a keyed TBC $\widetilde{E}_K$ is denoted by $\Pi[\widetilde{E}_K]$ and is a pair of encryption and decryption algorithms $(\Pi.\mathsf{Enc}[\widetilde{E}_K], \Pi.\mathsf{Dec}[\widetilde{E}_K])$. $\mathcal{K}, \mathcal{N}, \mathcal{M}, \mathcal{C}, \mathcal{A}$ and $\mathcal{T}$ are the sets of keys, nonces, plaintexts, ciphertexts, associated data (AD) and tags of the nAEAD scheme, respectively. In this paper, the key space of an nAEAD scheme is equal to that of the underlying TBC. The encryption algorithm takes a nonce $N \in \mathcal{N}$, AD $A \in \mathcal{A}$, and a plaintext $M \in \mathcal{M}$, and returns, deterministically, a pair of a ciphertext $C \in \mathcal{C}$

and a tag $T \in \mathcal{T}$. The decryption algorithm takes a tuple $(N, A, C, T) \in \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T}$, and returns, deterministically, either the distinguished invalid (reject) symbol $\perp \notin \mathcal{M}$ or a plaintext $M \in \mathcal{M}$. We require $|\Pi.\mathsf{Enc}[\widetilde{E}_K](N, A, M)| = |\Pi.\mathsf{Enc}[\widetilde{E}_K](N, A, M')|$ when these outputs are strings and $|M| = |M'|$.

We follow the security definition of an nAEAD scheme in [26, 33] that is the indistinguishability between $\Pi[\widetilde{E}_K] = (\Pi.\mathsf{Enc}[\widetilde{E}_K], \Pi.\mathsf{Dec}[\widetilde{E}_K])$ and $(\$, \perp)$, where $\$$ is a random-bits oracle that has the same interface as $\Pi.\mathsf{Enc}[\widetilde{E}_K]$ and for a query $(N, A, M)$ returns a random bit string of length $|\Pi.\mathsf{Enc}[\widetilde{E}_K](N, A, M)|$; $\perp$ is an oracle that returns the reject symbol $\perp$ for any query. In the nAEAD-security game, first an adversary $\mathbf{A}$ interacts with either $\Pi[\widetilde{E}_K]$ or $(\$, \perp)$, and then returns a decision bit $y \in \{0, 1\}$. For an nAEAD scheme $\Pi[\widetilde{E}_K]$, the nAEAD-security advantage function of an adversary $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\mathsf{naead}}_{\Pi[\widetilde{E}_K]}(\mathbf{A}) = \Pr[K \xleftarrow{\$} \mathcal{K}; \mathbf{A}^{\Pi[\widetilde{E}_K]} = 1] - \Pr[\mathbf{A}^{\$,\perp} = 1] \ ,$$

where the probabilities are taken over $K, \$$ and $\mathbf{A}$. We demand that $\mathbf{A}$ is nonce-respecting (all nonces in encryption queries are distinct), that $\mathbf{A}$ never asks a trivial decryption query $(N, A, C, T)$, i.e., there is a prior encryption query $(N, A, M)$ with $(C, T) = \Pi.\mathsf{Enc}[\widetilde{E}_K](N, A, M)$, and that $\mathbf{A}$ never repeats a query. Through this paper, the world with $\Pi[\widetilde{E}_K]$ is called "real world," and the world with $(\$, \perp)$ is called " ideal world. " Queries to $\Pi.\mathsf{Enc}[\widetilde{E}_K]/\$$ are called "encryption queries," and queries to $\Pi.\mathsf{Dec}[\widetilde{E}_K]/\perp$ are called "decryption queries."

The maximum over all adversaries, running in time at most $t$ and making at most $q_{\mathcal{E}}$ encryption queries and $q_{\mathcal{D}}$ decryption queries of $\sigma$ the total number of TBC calls invoked by all queries, is denoted by

$$\mathbf{Adv}^{\mathsf{naead}}_{\Pi[\widetilde{E}_K]}((q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma), t) := \max_{\mathbf{A}} \mathbf{Adv}^{\mathsf{naead}}_{\Pi[\widetilde{E}_K]}(\mathbf{A}) \ .$$

When an adversary is a computationally unbounded algorithm, the time $t$ is disregarded.

## 3 FBAE: TBC-based Feedback Mode

We design a TBC-based nAEAD scheme, basing on the iCOFB design approach.

### 3.1 Brief Overview of iCOFB Design and Security

iCOFB given in [10] is a tweakable random-function (TRF)-based nAEAD scheme and is designed so that an extra state beyond the TRF size is not required. Let $\ell_{\mathsf{max}}$ be the maximum length of ciphertext blocks, and $R : (\mathcal{N} \times \mathcal{A} \times [\ell_{\mathsf{max}} + 1] \times (1)) \times \{0,1\}^b \to \{0,1\}^b$ be a TRF, where tweak elements are a nonce, AD, a counter and a domain separation. Let $\rho : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^b \times \{0,1\}^b$ be a feedback function that takes a $b$-bit TRF output and a $b$-bit plaintext block, and outputs a $b$-bit TRF input and a $b$-bit ciphertext block. Figure 1 shows the encryption and decryption procedures of iCOFB with three plaintext/ciphertext blocks.

In order for iCOFB to become lightweight, the feedback function $\rho$ should be lightweight. [10] considers a linear function, thus $\rho$ is expressed by a $2b \times 2b$ binary matrix:

$$\rho(Y_i, M_i) = \begin{pmatrix} X_{i+1} \\ C_i \end{pmatrix} = \begin{pmatrix} E_{1,1} & E_{1,2} \\ E_{2,1} & E_{2,2} \end{pmatrix} \begin{pmatrix} Y_i \\ M_i \end{pmatrix}$$

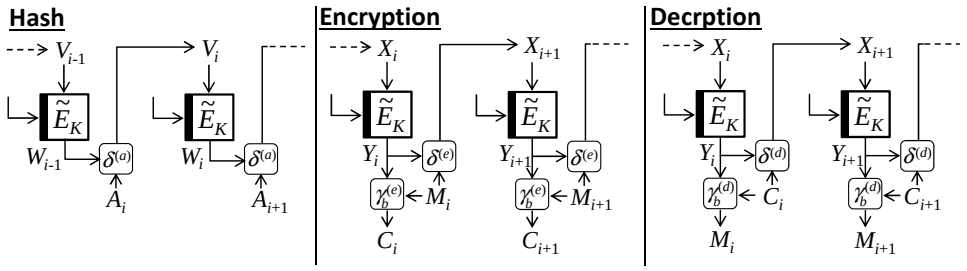


**Fig. 2.** Core Procedures of AD Processing (Hash), Encryption and Decryption. $A_i$ is an $i$-th AD block. $M_i$ is an $i$-th plaintext block. $C_i$ is an $i$-th ciphertext block. Tweaks are omitted.

where $E_{i,j}$'s are $b \times b$ binary matrices. For the decryption of iCOFB, the feedback function $\rho'$ is also expressed by a $2b \times 2b$ binary matrix:

$$\rho'(Y_i, C_i) = \begin{pmatrix} X_{i+1} \\ M_i \end{pmatrix} = \begin{pmatrix} D_{1,1} & D_{1,2} \\ D_{2,1} & D_{2,2} \end{pmatrix} \begin{pmatrix} Y_i \\ C_i \end{pmatrix}$$

where $D_{i,j}$'s are $b \times b$ binary matrices. For the correctness of iCOFB, [10] chooses the feedback function $\rho'$ with the following conditions: $E_{2,2}$ is invertible; $D_{1,1} = E_{1,1} + E_{1,2}E_{2,2}^{-1}E_{2,1}$; $D_{1,2} = E_{1,2}$; $D_{2,1} = E_{2,2}^{-1}E_{2,1}$; $D_{2,2} = E_{2,2}^{-1}$.

Regarding the security of iCOFB, they show the following theorem.

**Theorem 1.** *If the feedback function $\rho$ satisfies the conditions **(A1)** $E_{2,1}$ is invertible; **(A2)** $D_{1,2}$ is invertible; **(A3)** $D_{1,1}$ is invertible, then for any adversary $\mathbf{A}$ making at most $q_{\mathcal{D}}$ decryption queries of plaintext length at most $\ell_{\mathsf{max}}$ blocks,*

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{iCOFB}[R]}(\mathbf{A}) \leq \frac{q_{\mathcal{D}}(\ell_{\mathsf{max}} + 1)}{2^b} \quad .$$

### 3.2 FBAE: Design Principle and Specification

We design FBAE, a TBC-based lightweight AEAD mode, by extending the idea of iCOFB to the TBC setting.

**Encryption/Decryption Procedures** In FBAE, a plaintext/ciphertext is partitioned into $b$-bit blocks, and as iCOFB, each block is processed by a TBC and feedback function $\rho/\rho'$. But more general functions than the linear feedback functions are considered.

- The feedback function in the encryption is composed of the following two functions: for an integer $0 < l \leq b$,
  - $\gamma_l^{(e)} : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$ defines a ciphertext block $C_i \in \{0,1\}^l$ from a TBC output $Y_i \in \{0,1\}^l$ and a plaintext block $M_i \in \{0,1\}^l$ (thus $C_i = \gamma_l^{(e)}(Y_i, M_i)$), and
  - $\delta^{(e)} : \{0,1\}^b \times \{0,1\}^{\leq b} \to \{0,1\}^b$ defines a TBC input $X_{i+1} \in \{0,1\}^b$ from a TBC output $Y_i \in \{0,1\}^b$ and a plaintext block $M_i \in \{0,1\}^{\leq b}$ (thus $X_{i+1} = \delta^{(e)}(Y_i, M_i)$).

  The core procedure of the encryption of FBAE that uses these functions is given in Figure 2 (Center). Note that plaintext blocks except for the last block are of $l = b$, and the last block is of $l \leq b$.

– The feedback function in the decryption is composed of the following two functions: for an integer $0 < l \leq b$,
  - $\gamma_l^{(d)} : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l$ defines a plaintext block $M_i \in \{0,1\}^l$ from a TBC output $Y_i \in \{0,1\}^l$ and a ciphertext block $C_i \in \{0,1\}^l$ (thus $M_i = \gamma_l^{(d)}(Y_i, C_i)$), and
  - $\delta^{(d)} : \{0,1\}^b \times \{0,1\}^{\leq b} \to \{0,1\}^b$ defines a TBC input $X_{i+1} \in \{0,1\}^b$ from a TBC output $Y_i \in \{0,1\}^b$ and a ciphertext block $C_i \in \{0,1\}^{\leq b}$ (thus $X_{i+1} = \delta^{(d)}(Y_i, M_i)$).

  The core procedure of the decryption of FBAE that uses these functions is given in Figure 2 (Right). Note that ciphertext blocks except for the last block are of $l = b$, and the last block is of $l \leq b$.

**Hash Procedure (AD Processing)** In order to design a lightweight AEAD scheme, FBAE uses a fixed-tweak-length TBC, whereas iCOFB uses a variable-tweak-length TRF to take variable-length AD. Hence, we define additional procedure of processing variable-length AD. Similar to the encryption/decryption procedures, AD is partitioned into $b$-bit blocks and then AD blocks are processed by iterating a combination of a TBC and the following feedback function.

– $\delta^{(a)} : \{0,1\}^b \times \left(\{\lambda\} \cup \{0,1\}^{\leq b}\right) \to \{0,1\}^b$ defines a TBC input $V_i \in \{0,1\}^b$ from a TBC output $W_{i-1} \in \{0,1\}^b$ and an AD block $A_i \in \{\lambda\} \cup \{0,1\}^{\leq b}$ (thus $V_i = \delta^{(a)}(W_{i-1}, A_i)$).

Note that an empty AD block is appeared when AD is an empty string. The core procedure of processing an AD block is given in Figure 2 (Left).

**Tweak Function** Let $\ell_{\mathsf{max}}$ be the maximum block size of AD, plaintext and ciphertext. Regarding a tweak of the underlying TBC, we use the following tweak function:

– $f : [7] \times \mathcal{N} \times (\ell_{\mathsf{max}}) \to \mathcal{TW}$,

with the following condition:

– **B1**: for any $(i, N, j), (i', N', j') \in [7] \times \mathcal{N} \times (\ell_{\mathsf{max}})$ such that $(i, N, j) \neq (i', N', j')$,

$$f(i, N, j) \neq f(i', N', j').$$

The first element is used for distinguishing AD, plaintext/ciphertext and whether the last block is a full-bit one or not, which offers a distinct permutation between the hash procedure and the encryption/decryption, and which avoids additional TBC call when the last block is a full-bit one. The second element is a nonce, which offers a distinct permutation for each nonce thereby removing the birthday term regarding the number of queries. The third element is the current block number, which offers a distinct permutation for each block thereby removing the birthday term regarding the query length.

**Specification of FBAE** The specification of FBAE is given in Algorithm 1 and is shown in Figure 3. FBAE.Hash is the hash procedure, FBAE.Enc is the encryption, and FBAE.Dec is the decryption.

For the correctness of FBAE, the following conditions are required. Let $l$ be an integer such that $0 < l \leq b$.

– **B2**: for any $Y \in \{0,1\}^l$, $\gamma_l^{(e)}(Y, \cdot)$ is bijective and $\gamma_l^{(d)}(Y, \cdot)$ is the inverse of $\gamma_l^{(e)}(Y, \cdot)$, i.e., $M = \gamma_l^{(d)}(Y, \gamma_l^{(e)}(Y, M))$ for any $M \in \{0,1\}^l$.
– **B3**: for any $M \in \{0,1\}^l, Y \in \{0,1\}^b$, $\delta^{(e)}(Y, M) = \delta^{(d)}(Y, \gamma_l^{(e)}(\mathsf{msb}_l(Y), M))$.

**Algorithm 1** FBAE

**Encryption** $\mathsf{FBAE.Enc}[\widetilde{E}_K](N, A, M)$

1: $X_1 \leftarrow \mathsf{FBAE.Hash}[\widetilde{E}_K](A)$
2: **if** $A \neq \lambda \wedge |A| \mod b = 0$ **then** $x \leftarrow 2$; **else** $x \leftarrow 3$
3: **if** $M = \lambda$ **then** $\ell \leftarrow 0$; goto step 8
4: $M_1, \ldots, M_\ell \xleftarrow{b} M$
5: **for** $i = 1, \ldots, \ell$ **do**
6: $\quad Y_i \leftarrow \widetilde{E}_K^{f(x,N,i)}(X_i)$; $C_i \leftarrow \gamma^{(e)}_{|M_i|}(\mathsf{msb}_{|M_i|}(Y_i), M_i)$; $X_{i+1} \leftarrow \delta^{(e)}(Y_i, M_i)$
7: **end for**
8: **if** $M \neq \lambda \wedge |M| \mod b = 0$ **then** $y \leftarrow x + 2$; **else** $y \leftarrow x + 4$
9: $S \leftarrow X_{\ell+1}$; $T \leftarrow \mathsf{msb}_\tau\left(\widetilde{E}_K^{f(y,N,\ell)}(S)\right)$; $C \leftarrow C_1 \| \cdots \| C_\ell$
10: **return** $(C, T)$

**Decryption** $\mathsf{FBAE.Dec}[\widetilde{E}_K](N, A, C, T)$

1: $X_1 \leftarrow \mathsf{FBAE.Hash}[\widetilde{E}_K](A)$
2: **if** $A \neq \lambda \wedge |A| \mod b = 0$ **then** $x \leftarrow 2$; **else** $x \leftarrow 3$
3: **if** $C = \lambda$ **then** $\ell \leftarrow 0$; goto step 8
4: $C_1, \ldots, C_\ell \xleftarrow{b} C$
5: **for** $i = 1, \ldots, \ell$ **do**
6: $\quad Y_i \leftarrow \widetilde{E}_K^{f(x,N,i)}(X_i)$; $M_i \leftarrow \gamma^{(d)}_{|C_i|}(\mathsf{msb}_{|C_i|}(Y_i), C_i)$; $X_{i+1} \leftarrow \delta^{(d)}(Y_i, C_i)$
7: **end for**
8: **if** $C \neq \lambda \wedge |C| \mod b = 0$ **then** $y \leftarrow x + 2$; **else** $y \leftarrow x + 4$
9: $S \leftarrow X_{\ell+1}$; $\hat{T} \leftarrow \mathsf{msb}_\tau\left(\widetilde{E}_K^{f(y,N,\ell)}(S)\right)$; $M \leftarrow M_1 \| \cdots \| M_\ell$
10: **if** $T = \hat{T}$ **then return** $M$; **else return** $\perp$

**Hash** $\mathsf{FBAE.Hash}[\widetilde{E}_K](A)$

1: $A_1, \ldots, A_a \xleftarrow{b} A$; $W_0 \leftarrow 0^b$
2: **for** $i = 1, \ldots, a-1$ **do** $V_i \leftarrow \delta^{(d)}(W_{i-1}, A_i)$; $W_i \leftarrow \widetilde{E}_K^{f(1,R,i)}(V_i)$
3: $V_a \leftarrow \delta^{(d)}(W_{a-1}, A_a)$; $H \leftarrow V_a$
4: **return** $H$

### 3.3 Conditions on $\gamma_l^{(e)}, \gamma_l^{(d)}, \delta^{(a)}, \delta^{(e)}, \delta^{(d)}$ for Achieving nAE-Security

In order to prove the nAE-security of FBAE, we introduce the following five conditions on $\gamma_l^{(e)}, \gamma_l^{(d)}, \delta^{(a)}, \delta^{(e)}, \delta^{(d)}$.

- **B4**: for any $M \in \{0,1\}^l$, $\gamma_l^{(e)}(\cdot, M)$ is bijective.
- **B5**: for any $C \in \{0,1\}^{\leq b}$, $\delta^{(d)}(\cdot, C)$ is bijective.
- **B6**: for any $C, C' \in \{0,1\}^{\leq b}$ and $Y, Y' \in \{0,1\}^b$,

$$\begin{aligned}\delta^{(e)}(Y, \gamma^{(d)}_{|C|}(\mathsf{msb}_{|C|}(Y), C)) = \delta^{(d)}(Y', C') \Rightarrow &(C = C' \wedge Y = Y') \vee \\ &(C \neq C' \wedge Y \neq Y') \vee \\ &(C \neq C' \wedge |C| = b \wedge |C'| < b \wedge Y = Y') \vee \\ &(C \neq C' \wedge |C| < b \wedge |C'| = b \wedge Y = Y').\end{aligned}$$

- **B7**: for any $A \in \{0,1\}^{\leq b}$, $\delta^{(a)}(\cdot, A)$ is bijective.

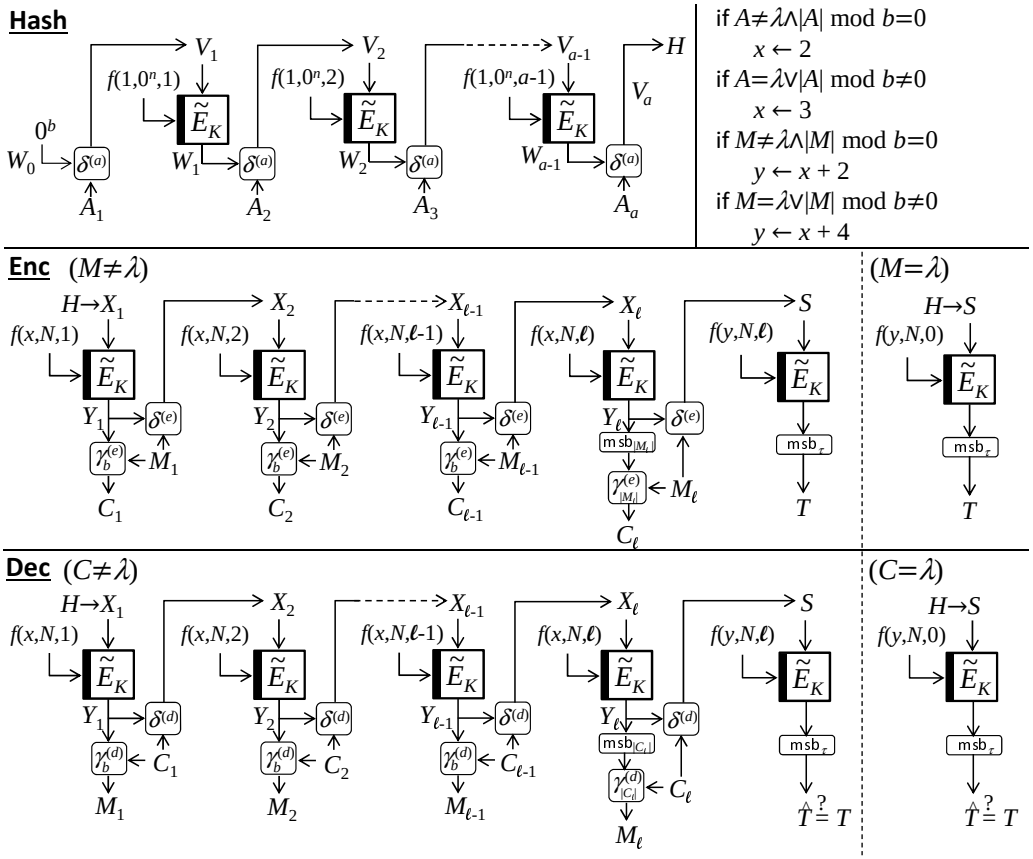


**Fig. 3.** FBAE. $A_1, \ldots, A_a \xleftarrow{b} A$. $M_1, \ldots, M_\ell \xleftarrow{b} M$ (in the encryption algorithm) and $C_1, \ldots, C_\ell \xleftarrow{b} C$ (in the decryption algorithm).

– **B8**: for any $A, A' \in \{\lambda\} \cup \{0,1\}^{\leq b}, W, W' \in \{0,1\}^b$,

$$\begin{aligned}\delta^{(a)}(W, A) = \delta^{(a)}(W, A') \Rightarrow &(A = A' \wedge W = W') \vee \\ &(A \neq A' \wedge W \neq W') \vee \\ &(A \neq A' \wedge |A| = b \wedge |A'| < b \wedge W = W').\end{aligned}$$

The condition **B4** ensures that for a plaintext block $M_i$ and a TBC output $Y_i$, if $Y_i$ is uniformly distributed over $\{0,1\}^b$, then so is the ciphertext block $C_i = \gamma_l^{(e)}(Y_i, M_i)$. The condition **B5** ensures that the internal state collision $\delta^{(e)}(Y_i, M_i) = \delta^{(d)}(Y_i', C_i')$ (between the encryption and decryption) depends on the randomness of the TBC output $Y_i'$. Thus, if the output is distributed over a set $\mathcal{X}$, then the collision probability can be at most $1/|\mathcal{X}|$. Similar to the condition **B5**, the condition **B7** ensures that in the procedure of processing AD blocks, the internal state collision $\delta^{(a)}(W_i, A_i) = \delta^{(a)}(W_i', A_i')$ depends on the randomness of the TBC output $W_i'$. The conditions **B5**, **B7** are used to upper bound the probability of forging a tag. The condition **B6** ensures that in the encryption and decryption procedures, no trivial collision occurs on the internal state values. Note that the condition **B6** tolerates an internal state collision from the

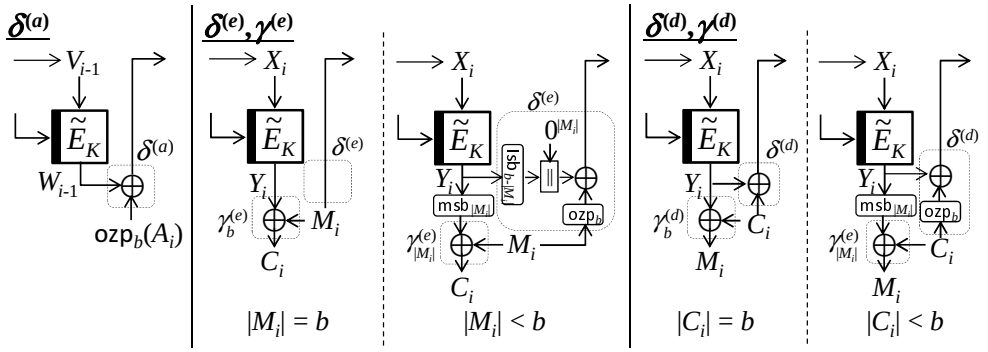


**Fig. 4.** Lightweight Instantiations of $\delta^{(a)}, \delta^{(e)}, \delta^{(d)}, \gamma^{(e)}, \gamma^{(d)}$.

conditions $(C \neq C' \wedge |C| = b \wedge |C'| < b \wedge Y = Y')$ and $(C \neq C' \wedge |C| < b \wedge |C'| = b \wedge Y = Y')$ but the first element of $f$ gets rid of the influence of the trivial collision. The condition **B6** is defined similarly.

It is easy to see that the classes of the functions $\gamma_l^{(e)}$, $\gamma_l^{(d)}$, $\delta^{(e)}$, $\delta^{(d)}$ with the conditions **B4**, **B5**, **B6** include the linear feedback function $\rho, \rho'$ with the conditions **A1**, **A2**, **A3**.

### 3.4 Lightweight Instantiations of $\gamma_l^{(e)}, \gamma_l^{(d)}, \delta^{(a)}, \delta^{(e)}, \delta^{(d)}, f$

In the section 5, we show that FBAE offers a lightweight AEAD scheme, combining with a lightweight TBC. In the implementation, the following lightweight functions are used.

$$\gamma_l^{(e)}(Y_i, M_i) = Y_i \oplus M_i, \text{ where } 0 < l \leq b, \text{ and } Y_i, M_i \in \{0,1\}^l$$
$$\gamma_l^{(d)}(Y_i, C_i) = Y_i \oplus C_i, \text{ where } 0 < l \leq b, \text{ and } Y_i, C_i \in \{0,1\}^l$$
$$\delta^{(e)}(Y_i, M_i) = \mathsf{ozp}_b(M_i) \oplus \left(0^{|M_i|} \| \mathsf{lsb}_{b-|M_i|}(Y_i)\right), \text{ where } Y_i \in \{0,1\}^b, M_i \in \{0,1\}^{\leq b}.$$
$$\delta^{(d)}(Y_i, C_i) = Y_i \oplus \mathsf{ozp}_b(C_i), \text{ where } Y_i \in \{0,1\}^b, C_i \in \{0,1\}^{\leq b}.$$
$$\delta^{(a)}(W_i, A_i) = W_i \oplus \mathsf{ozp}_b(A_i), \text{ where } Y_i \in \{0,1\}^b, A_i \in \{\lambda\} \cup \{0,1\}^{\leq b}.$$

These functions are shown in Figure 4. FBAE with the above functions is called PFB (Plaintext FeedBack). PFB is shown in Figure 7 in Appendix. It is easy to see that the above functions satisfy the conditions **B2**-**B8**.

The tweak function $f$, when $\mathcal{TW} := \{0,1\}^t$ and $\mathcal{N} := \{0,1\}^n$ such that $n + 3 + 1 \leq t$, is defined as

$$f(i, N, j) = (\mathsf{str}_3(i) \| N \| \mathsf{str}_{t-3-n}(j)),$$

which satisfies the condition **B1**.

## 4 nAEAD-Security of FBAE

The nAEAD-security bound of FBAE is given in the following theorem.

**Theorem 2.** *For* FBAE *with the conditions* ***B1***-***B8****, we have*

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{E}_K]}((q_\mathcal{E}, q_\mathcal{D}, \sigma), t) \leq \frac{q_\mathcal{D}}{2^\tau - 1/2^{b-\tau}} + \frac{q_\mathcal{D}}{2^b - 1} + \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma, t + O(\sigma)) \ .$$

The proof is given in the following subsections.

### 4.1 Replacing the Keyed TBC $E_K$ with a TRP $\widetilde{\mathbf{P}}$

The keyed TBC $\widetilde{E}_K$ for $K \xleftarrow{\$} \mathcal{K}$ is replaced with a TRP $\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right)$. By the replacement, we have

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{E}_K]}((q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma), t) \leq \mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{P}]}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma) + \mathbf{Adv}^{\mathsf{tprp}}_{\widetilde{E}_K}(\sigma, t + O(\sigma)) \ . \tag{1}$$

Hereafter, $\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{\mathbf{P}}]}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma)$, the nAEAD-advantage of $\mathsf{FBAE}[\widetilde{P}]$ is upper bounded, where an adversary is a computationally unbounded algorithm and the complexity is solely measured by the numbers of queries. Without loss of generality, an adversary is deterministic.

### 4.2 Upper Bounding $\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{P}]}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma)$

Firstly, a forgery event in the real world is defined.

$$\mathsf{forge} \Leftrightarrow \exists i \in [q_{\mathcal{D}}] \text{ s.t. at the } i\text{-th decryption query, } \perp \text{ is returned.}$$

Then, for any adversary $\mathbf{A}$,

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{P}]}(\mathbf{A}) = \Pr\left[\widetilde{\mathbf{P}} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right); \mathbf{A}^{\mathsf{FBAE}[\widetilde{P}]} = 1\right] - \Pr\left[\mathbf{A}^{\$,\perp} = 1\right] \\
&\leq \Pr[\mathsf{forge}] + \Pr\left[\widetilde{\mathbf{P}} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right); \mathbf{A}^{\mathsf{FBAE}[\widetilde{P}]} = 1 | \neg\mathsf{forge}\right] - \Pr\left[\mathbf{A}^{\$,\perp} = 1\right] \ .
\end{aligned} \tag{2}$$

In Subsection 4.3, $\Pr\left[\widetilde{\mathbf{P}} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right); \mathbf{A}^{\mathsf{FBAE}[\widetilde{P}]} = 1 | \neg\mathsf{forge}\right] - \Pr\left[\mathbf{A}^{\$,\perp} = 1\right]$ is analyzed. In Subsection 4.4, $\Pr[\mathsf{forge}]$ is analyzed. Putting the upper bounds (4), (5) into (2) gives

$$\mathbf{Adv}^{\mathsf{naead}}_{\mathsf{FBAE}[\widetilde{P}]}(q_{\mathcal{E}}, q_{\mathcal{D}}, \sigma) \leq \frac{q_{\mathcal{D}}}{2^\tau - 1/2^{b-\tau}} + \frac{q_{\mathcal{D}}}{2^b - 1} \ , \tag{3}$$

and putting the above upper bound into (1) gives that in Theorem 2.

### 4.3 Analysis of $\Pr\left[\widetilde{\mathbf{P}} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right); \mathbf{A}^{\mathsf{FBAE}[\widetilde{P}]} = 1 | \neg\mathsf{forge}\right] - \Pr\left[\mathbf{A}^{\$,\perp} = 1\right]$

In the real world, the condition **B1** of the tweak function $f$ ensures that all tweaks of $\widetilde{P}$ defined by encryption queries are distinct. Hence, the output blocks of $\widetilde{P}$ are chosen independently and uniformly at random from $\{0,1\}^b$. By the condition **B4**, all ciphertext blocks $C_i$ defined by encryption queries are independently and uniformly distributed over $\{0,1\}^{|C_i|}$, and thus are indistinguishable from those defined by \$. By $\neg\mathsf{forge}$, all outputs of $\mathsf{FBAE.Dec}[\widetilde{\mathbf{P}}]$ are $\perp$. Hence, we have

$$\Pr\left[\widetilde{P} \xleftarrow{\$} \widetilde{\mathsf{Perm}}\left(\mathcal{TW}, \{0,1\}^b\right); \mathbf{A}^{\Pi[\widetilde{P}]} = 1 | \neg\mathsf{forge}\right] - \Pr\left[\mathbf{A}^{\$,\perp} = 1\right] = 0 \ . \tag{4}$$

### 4.4 Analysis of Pr[forge]

In the following analysis, without loss of generality, an adversary $\mathbf{A}$ aborts after $\mathsf{forge}$ occurs. Let $\mathsf{forge}_i$ be an event that at the $i$-th decryption query $\mathsf{forge}$ occurs (thus $\mathsf{forge}_i$ occurs as long as $\mathsf{forge}_1 \vee \mathsf{forge}_2 \vee \cdots \vee \mathsf{forge}_{i-1}$ does not occur). Thus,

$$\Pr[\mathsf{forge}] \leq \sum_{i=1}^{q_{\mathcal{D}}} \Pr[\mathsf{forge}_i] \ .$$

Next, $\Pr[\mathsf{forge}_i]$ is upper bounded, wherein $i \in [q_{\mathcal{D}}]$. A value/variable $V$ defined at the $i$-th decryption query, except for the lengths $a$ and $\ell$, is denoted by $V^{(d)}$. The lengths $a$ and $\ell$ are denoted by $a_d$ and $\ell_d$, respectively. Similarly, for an encryption query $(N^{(e)}, A^{(e)}, M^{(e)})$, a value/variable $V$ corresponding with the encryption query, except for the lengths $a$ and $\ell$, is denoted by $V^{(e)}$. The lengths $a$ and $\ell$ are denoted by $a_e$ and $\ell_e$, respectively. In this analysis, we consider the following types of decryption query.

- $\mathsf{Type\text{-}1}$: For any previous encryption query $(N^{(e)}, A^{(e)}, M^{(e)})$,

$$N^{(e)} \neq N^{(d)} \vee y^{(e)} \neq y^{(d)} \vee \ell_e \neq \ell_d.$$

- $\mathsf{Type\text{-}2}$: For some previous encryption query $(N^{(e)}, A^{(e)}, M^{(e)})$,

$$N^{(e)} = N^{(d)} \wedge y^{(e)} = y^{(d)} \wedge \ell_e = \ell_d.$$

Then,

$$\begin{aligned}\Pr[\mathsf{forge}_i] &= \Pr[\mathsf{forge}_i \wedge \mathsf{Type\text{-}1}] + \Pr[\mathsf{forge}_i \wedge \mathsf{Type\text{-}2}] \\ &= \Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}1}\right] \cdot \Pr\left[\mathsf{Type\text{-}1}\right] + \Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}2}\right] \cdot \Pr\left[\mathsf{Type\text{-}2}\right] \\ &\leq \max\left\{\Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}1}\right], \Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}2}\right]\right\} \ .\end{aligned}$$

In the subsection 4.5, $\Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}1}\right]$ is analyzed, and in the subsection 4.6, $\Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}2}\right]$ is analyzed. The upper bounds (6), (7) give

$$\Pr[\mathsf{forge}_i] \leq \frac{1}{2^\tau - 1/2^{b-\tau}} + \frac{1}{2^b - 1} \ .$$

Thus we have

$$\Pr[\mathsf{forge}] \leq q_{\mathcal{D}} \cdot \left(\frac{1}{2^\tau - 1/2^{b-\tau}} + \frac{1}{2^b - 1}\right) = \frac{q_{\mathcal{D}}}{2^\tau - 1/2^{b-\tau}} + \frac{q_{\mathcal{D}}}{2^b - 1} \ . \tag{5}$$

### 4.5 Analysis of Pr [$\mathsf{forge}_i$|Type-1]

Under the $\mathsf{Type\text{-}1}$ decryption query and by the condition $\mathbf{B1}$, the tweak $f(y^{(d)}, N^{(d)}, \ell_d)$, with which the TRP defines the tag $\hat{T}^{(d)}$, is distinct from all tweaks defined by the previous encryption queries, and is distinct from other tweaks defined by the decryption query. Hence, $\hat{T}^{(d)}$ is uniformly distributed over $\{0,1\}^\tau$ and independent of the TRP outputs defined by the previous encryption queries and of other TRP outputs defied by the decryption query. Thus, we have

$$\Pr\left[\mathsf{forge}_i | \mathsf{Type\text{-}1}\right] \leq \frac{1}{2^\tau} \ . \tag{6}$$

### 4.6 Analysis of $\Pr[\mathsf{forge}_i|\mathsf{Type\text{-}2}]$

$\Pr\left[\hat{T}^{(d)} = T^{(d)} \middle| S^{(d)} \neq S^{(e)} \wedge \mathsf{Type\text{-}2}\right]$ and $\Pr\left[S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2}\right]$ are upper bounded, since

$$\begin{aligned}
\Pr\left[\mathsf{forge}_i|\mathsf{Type\text{-}2}\right] &= \Pr\left[\hat{T}^{(d)} = T^{(d)} \wedge S^{(d)} \neq S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \\
&\quad + \Pr\left[\hat{T}^{(d)} = T^{(d)} \wedge S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \\
&= \Pr\left[\hat{T}^{(d)} = T^{(d)} \middle| \mathsf{Type\text{-}2} \wedge S^{(d)} \neq S^{(e)}\right] \cdot \Pr\left[S^{(d)} \neq S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \\
&\quad + \Pr\left[\hat{T}^{(d)} = T^{(d)} \middle| \mathsf{Type\text{-}2} \wedge S^{(d)} = S^{(e)}\right] \cdot \Pr\left[S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \\
&\leq \Pr\left[\hat{T}^{(d)} = T^{(d)} \middle| \mathsf{Type\text{-}2} \wedge S^{(d)} \neq S^{(e)}\right] + \Pr\left[S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \ .
\end{aligned}$$

The upper bounds (8), (11) give

$$\Pr\left[\mathsf{forge}_i|\mathsf{Type\text{-}2}\right] \leq \frac{1}{2^\tau - 1/2^{b-\tau}} + \frac{1}{2^b - 1} \ . \tag{7}$$

**Upper Bounding $\Pr\left[\hat{\boldsymbol{T}}^{(\boldsymbol{d})} = \boldsymbol{T}^{(\boldsymbol{d})} \middle| \mathsf{Type\text{-}2} \wedge \boldsymbol{S}^{(\boldsymbol{d})} \neq \boldsymbol{S}^{(\boldsymbol{e})}\right]$** For the Type-2 decryption query, by $S^{(d)} \neq S^{(e)}$ and $f(y^{(e)}, N^{(e)}, \ell_e) = f(y^{(d)}, N^{(d)}, \ell_d)$ (the tweaks are the same), the output of the last TRP call by the decryption query is chosen uniformly at random from $\{0,1\}^b \backslash \{\widetilde{P}^{f(y^{(e)}, N^{(e)}, \ell_e)}(S^{(e)})\}$. We thus have

$$\Pr\left[\hat{T}^{(d)} = T^{(d)} \middle| \mathsf{Type\text{-}2} \wedge S^{(d)} \neq S^{(e)}\right] \leq \frac{2^{b-\tau}}{2^b - 1} = \frac{1}{2^\tau - 1/2^{b-\tau}} \ . \tag{8}$$

**Upper Bounding $\Pr\left[\boldsymbol{S}^{(\boldsymbol{d})} = \boldsymbol{S}^{(\boldsymbol{e})} \middle| \mathsf{Type\text{-}2}\right]$** The condition of the Type-2 decryption query, $y^{(e)} = y^{(d)}$, is satisfied if and only if

$$\left(\left|A^{(d)}_{a_d}\right| = \left|A^{(e)}_{a_e}\right| = b\right) \vee \left(\left|A^{(d)}_{a_d}\right| < b \wedge \left|A^{(e)}_{a_e}\right| < b\right) \text{ and} \tag{9}$$

$$\left(\left|M^{(d)}_{\ell_d}\right| = \left|M^{(e)}_{\ell_e}\right| = b\right) \vee \left(\left|M^{(e)}_{\ell_e}\right| < b \wedge \left|M^{(d)}_{\ell_d}\right| < b\right). \tag{10}$$

Under the Type-2 decryption query, $\ell_e = \ell_d$ is satisfied. Let

$$I(A^{(d)}, A^{(e)}) = \left\{i \in [a_d] \middle| A^{(d)}_i \neq A^{(e)}_i\right\} \text{ and } I(C^{(d)}, C^{(e)}) = \left\{i \in [\ell_d] \middle| C^{(d)}_i \neq C^{(e)}_i\right\}$$

be sets of distinct blocks obtained from $(A^{(d)}, A^{(e)})$ and $(M^{(d)}, M^{(e)})$, respectively, where for $a_d < i$, $A^{(e)}_i := \lambda$.

Then,

$$\begin{aligned}\Pr\left[S^{(d)}=S^{(e)}\middle|\mathsf{Type\text{-}2}\right] &= \Pr\left[S^{(d)}=S^{(e)}\wedge|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &\quad+\Pr\left[S^{(d)}=S^{(e)}\wedge|I(C^{(d)},C^{(e)})|\geq 1\middle|\mathsf{Type\text{-}2}\right]\\ &\leq \underbrace{\Pr\left[S^{(d)}=S^{(e)}\wedge|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]}_{=:p_1}\\ &\quad+\underbrace{\Pr\left[S^{(d)}=S^{(e)}\middle|\mathsf{Type\text{-}2}\wedge|I(C^{(d)},C^{(e)})|\geq 1\right]}_{=:p_2}\\ &\quad\cdot\Pr\left[|I(C^{(d)},C^{(e)})|\geq 1\middle|\mathsf{Type\text{-}2}\right]\ .\end{aligned}$$

Regarding $p_1$, by the condition **B6**, for $Y,Y'\in\{0,1\}^b$ and $C\in\{0,1\}^l$, $\delta^{(e)}(Y,\gamma^{(d)}(Y,C))=\delta^{(d)}(Y',C)\Rightarrow Y=Y'$, and thus by $|I(C^{(d)},C^{(e)})|=0$,

$$S^{(d)}=S^{(e)}\Rightarrow H^{(d)}=H^{(e)}\ .$$

Hence, we have

$$\begin{aligned}p_1 &= \Pr\left[H^{(d)}=H^{(e)}\wedge|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &= \Pr\left[H^{(d)}=H^{(e)}\wedge a_e=a_d\wedge|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &\quad+\Pr\left[H^{(d)}=H^{(e)}\wedge a_e\neq a_d\wedge|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &= \underbrace{\Pr\left[H^{(d)}=H^{(e)}\middle|\mathsf{Type\text{-}2}\wedge a_e=a_d\wedge|I(C^{(d)},C^{(e)})|=0\right]}_{=:p_{1,1}}\\ &\quad\cdot\Pr\left[a_e=a_d\middle|\mathsf{Type\text{-}2}\wedge|I(C^{(d)},C^{(e)})|=0\right]\cdot\Pr\left[|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &\quad+\underbrace{\Pr\left[H^{(d)}=H^{(e)}\middle|\mathsf{Type\text{-}2}\wedge a_e\neq a_d\wedge|I(C^{(d)},C^{(e)})|=0\right]}_{=:p_{1,2}}\\ &\quad\cdot\Pr\left[a_e\neq a_d\middle|\mathsf{Type\text{-}2}\wedge|I(C^{(d)},C^{(e)})|=0\right]\cdot\Pr\left[|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &\leq \max\left\{p_{1,1},p_{1,2}\right\}\cdot\Pr\left[|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\ .\end{aligned}$$

Using these upper bounds, we have

$$\begin{aligned}\Pr\left[S^{(d)}=S^{(e)}\middle|\mathsf{Type\text{-}2}\right] &\leq \max\left\{p_{1,1},p_{1,2}\right\}\cdot\Pr\left[|I(C^{(d)},C^{(e)})|=0\middle|\mathsf{Type\text{-}2}\right]\\ &\quad+p_2\cdot\Pr\left[|I(C^{(d)},C^{(e)})|\geq 1\middle|\mathsf{Type\text{-}2}\right]\\ &\leq \max\left\{p_{1,1},p_{1,2},p_2\right\}\ .\end{aligned}$$

$p_{1,1},p_{1,2},p_2$ are upper bounded below.

- $p_{1,1}=\Pr\left[H^{(d)}=H^{(e)}\middle|\mathsf{Type\text{-}2}\wedge a_e=a_d\wedge|I(C^{(d)},C^{(e)})|=0\right]$ is upper bounded. Let $i=\max I(A^{(d)},A^{(e)})$. Then, by the condition **B8**,

$$H^{(e)}=H^{(d)}\Rightarrow V_i^{(e)}=V_i^{(d)}.$$

If $i = 1$, then

$$H^{(e)} = \delta^{(a)}(0^b, A_1^{(e)}) \text{ and } H^{(d)} = \delta^{(a)}(0^b, A_1^{(d)}) \ .$$

On the other hand, as $A_1^{(e)} \neq A_1^{(d)}$, by the condition **B8** with the conditions in (9) (thus the last condition in **B8**, $(A \neq A' \wedge |A| = b \wedge |A'| < b \wedge W = W')$, can be removed), we have

$$H^{(e)} = \delta^{(a)}(0^b, A_1^{(e)}) \neq \delta^{(a)}(0^b, A_1^{(d)}) = H^{(d)}.$$

Hence, we consider the case where $i \geq 2$. Then,

$$V_i^{(e)} = \delta^{(a)}\left(W_{i-1}^{(e)}, A_i^{(e)}\right) = \delta^{(a)}\left(W_{i-1}^{(d)}, A_i^{(d)}\right) = V_i^{(d)} \ .$$

By $A_i^{(d)} \neq A_i^{(e)}$ and the condition **B8** with the conditions in (9), in order to satisfy the above equation, $W_{i-1}^{(d)} \neq W_{i-1}^{(e)}$ should be satisfied. As $W_{i-1}^{(d)}$ is chosen uniformly at random from $\{0,1\}^b \backslash \left\{W_{i-1}^{(e)}\right\}$ and $\delta^{(d)}\left(\cdot, A_i^{(d)}\right)$ is bijective from the condition **B7**, we have

$$p_{1,1} \leq \frac{1}{2^b - 1} \ .$$

– $p_{1,2} = \Pr\left[H^{(d)} = H^{(e)} \middle| \mathsf{Type\text{-}2} \wedge a_e \neq a_d \wedge |I(C^{(d)}, C^{(e)})| = 0\right]$ is upper bounded. Thus, the following equation is considered.

$$H^{(d)} = \delta^{(a)}\left(W_{a_d-1}^{(d)}, A_{a_d}^{(d)}\right) = \delta^{(a)}\left(W_{a_e-1}^{(e)}, A_{a_e}^{(e)}\right) = H^{(e)} \ .$$

By $a_e \neq a_d$, the tweaks corresponding with the TRP outputs $W_{a_d-1}^{(d)}$ and $W_{a_e-1}^{(e)}$ are distinct. Thus, $W_{a_d-1}^{(d)}$ and $W_{a_e-1}^{(e)}$ are independently chosen, and at least one of them is chosen uniformly at random from $\{0,1\}^b$. (Note that for $x \in \{a, e\}$ if $a_x = 1$ then $H^{(x)} = \delta^{(a)}\left(0^b, A_1^{(x)}\right)$ which is a constant.) By the condition **B7**, at least one of $\delta^{(a)}\left(W_{a_d-1}^{(d)}, A_{a_d}^{(d)}\right)$ and $\delta^{(a)}\left(W_{a_e-1}^{(e)}, A_{a_e}^{(e)}\right)$ are uniformly distributed over $\{0,1\}^b$. Hence, we have

$$p_{1,2} \leq \frac{1}{2^b} \ .$$

– $p_2 = \Pr\left[S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2} \wedge |I(C^{(d)}, C^{(e)})| \geq 1\right]$ is upper bounded. Let $i = \max I(C^{(d)}, C^{(e)})$. Note that under the $\mathsf{Type\text{-}2}$ decryption query, $\ell_e = \ell_d$ is satisfied. Then by the condition **B6**,

$$\begin{aligned} S_1^{(d)} = S_1^{(e)} &\Leftrightarrow X_{i+1}^{(d)} = X_{i+1}^{(e)} \\ &\Leftrightarrow \delta^{(d)}\left(Y_i^{(d)}, C_i^{(d)}\right) = \delta^{(e)}\left(Y_i^{(e)}, M_i^{(e)}\right), \end{aligned}$$

where $M_i^{(e)} = \gamma^{(d)}(\mathsf{msb}_{|C_i^{(e)}|}(Y_i^{(e)}), C_i^{(e)})$. By the condition **B6** with (10) and $C_i^{(d)} \neq C_i^{(e)}$, $Y_i^{(d)} \neq Y_i^{(e)}$ is satisfied, thus $X_i^{(d)} \neq X_i^{(e)}$. Hence,

$$p_2 \leq \Pr\left[\delta^{(e)}\left(Y_i^{(e)}, M_i^{(e)}\right) = \delta^{(d)}\left(Y_i^{(d)}, C_i^{(d)}\right) \middle| \mathsf{Type\text{-}2} \wedge X_i^{(d)} \neq X_i^{(e)} \wedge |I(C^{(d)}, C^{(e)})| \geq 1\right] \ .$$

By $X_i^{(d)} \neq X_i^{(e)}$, $Y_i^{(d)}$ is chosen uniformly at random from $\{0,1\}^b \backslash \{Y_i^{(e)}\}$. By the condition **B5**, $\delta^{(d)}\left(C_i^{(d)}, Y_i^{(d)}\right)$ is uniformly distributed over $\{0,1\}^b \backslash \{\delta^{(e)}\left(C_i^{(e)}, Y_i^{(e)}\right)\}$, and we thus have

$$p_2 \leq \frac{1}{2^b - 1} \ .$$

The above upper bounds give

$$\Pr\left[S^{(d)} = S^{(e)} \middle| \mathsf{Type\text{-}2}\right] \leq \frac{1}{2^b - 1} \ . \tag{11}$$

## 5 Implementation

The performance of PFB is evaluated through concrete hardware implementations. For the lightweight TBC, we use a variant of SKINNY having the 64-bit block length and 192-bit tweakey, i.e., SKINNY-64-192 [6]. Its performance is compared with the state-of-the-art alternative having the same level of security: SAEB [25] instantiated with the lightweight block cipher GIFT-128-128 [4]. In the following, SKINNY-64-192 and GIFT-128-128 are simply referred to as SKINNY and GIFT. In addition, a mode of operation M instantiated with a primitive P is described as M[P].

*Design Policy* For a fair comparison, PFB[SKINNY] and SAEB[GIFT] are implemented under the same design policy. They are designed as co-processors aiming at accelerating the main time-consuming part of AD processing, encryption, and decryption. Meanwhile, the co-processors expect an external controller for handling special cases such as padding and the final-block processing. In order to avoid a hidden cost, the designs hold a key, nonce, and tweak during their lifetimes. In other words, there is no need for storing them in external registers and feeding them multiple times. This policy affect the implementation of on-the-fly key scheduling as we will see in the next section. The circuit area has the highest priority in optimization. The designs are described by a hardware description language (HDL) in register-transfer level (RTL). We do not make netlist-level optimization except scan flip-flops commonly used for compact implementations [24]; the standard cells for scan flip-flops are explicitly instantiated in HDL. For SCA-protected implementations, we consider TI secure up to the first-order attacks.

### 5.1 PFB[SKINNY]

SKINNY uses three distinct 64-bit states namely **TK1**, **TK2**, and **TK3** for tweakey schedule. In this particular design, **TK3** stores a 64-bit tweak. The remaining **TK1** and **TK2** store a 128-bit secret key.

Fig. 5 shows the hardware architecture of PFB[SKINNY]. As shown in Fig. 5, PFB[SKINNY] is realized as a thin wrapper of the SKINNY implementation; the additional components are 4-bit XOR, selector, and AND gate only.

The SKINNY implementation follows the conventional nibble-serial architecture [6], but the tweakey-schedule implementation is designed from scratch. The implementations called the **TK1**, **TK2**, and **TK3** arrays are based on a common architecture comprising an array of scan flip-flops and integrated on-the-fly key scheduling [24] as shown in Fig. 5. However, the changes made by the on-the-fly key scheduling should be reverted to begin the next TBC call without feeding the same key again. Since SKINNY schedules **TK1**, **TK2**, and **TK3** by a nibble permutation and a nibble-wise linear transformation for each round, we can obtain efficient inverse maps that revert the final tweakey state to the initial one. Such inverse maps are integrated to the **TK1**, **TK2**, and **TK3** arrays along with the forward on-the-fly scheduling.

Based on (3.4), the 64-bit tweak is given by $\mathsf{id}\|N\|\mathsf{ctr}$: a 3-bit number distinguishing the operations $\mathsf{id} = \mathsf{str}_3(i)$, 45-bit nonce $N$, and a current block number realized by a 16-bit counter $\mathsf{ctr} = \mathsf{str}_{16}(j)$. id and ctr are updated for each TBC call. For an efficient computation, the **TK3**

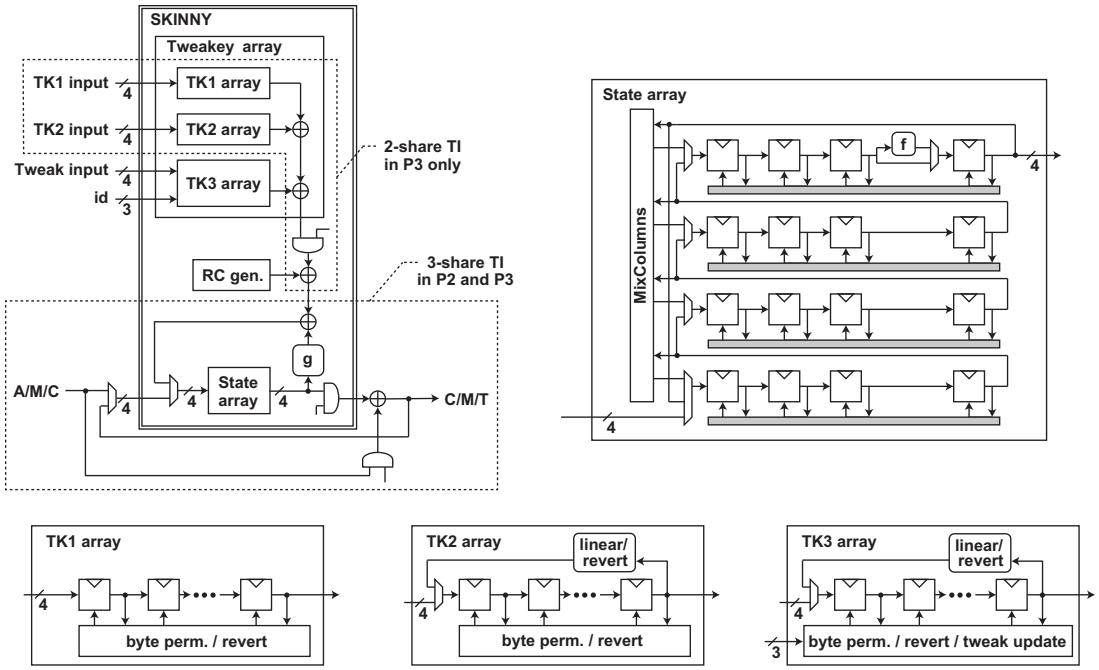


**Fig. 5.** PFB[SKINNY] hardware architecture.

array integrates the circuit for (i) changing id and (ii) incrementing and clearing the counter ctr. Using the above functionality, a user needs to feed $\mathsf{id}\|N\|\mathsf{ctr}$ only once for a given nonce $N$.

Single SKINNY round uses 16 cycles, and thus SKINNY comprising 40 rounds finishes in $16 \times 40 = 640$ cycles. We need an additional 1 cycle for updating a tweak stored in the **TK3** array for the next TBC call. As a result, a 64-bit message or ciphertext block is consumed in 641 cycles.

### 5.2 SAEB[GIFT]

Fig. 6 shows the hardware architecture of SAEB[GIFT]. The overall architecture is based on the conventional design [25], but the shift registers for synchronization are removed considering the design policy. It is also realized as a thin wrapper of the underlying GIFT implementation.

The GIFT implementation is based on the nibble-serial architecture [4], but the key array is redesigned to efficiently reverting the changes made by on-the-fly key scheduling. Similar to SKINNY, GIFT has a linear key scheduling algorithm, and thus we can obtain an efficient inverse map that revert the final key state to the initial one. The key array is designed with a 32-bit datapath to efficiently integrate the inverse key-schedule map (the function block labeled with "revert") as shown in Fig. 6.

The S-box is split into two stages namely $g$ and $f$ for TI following the conventional work [13]. Consequently, a single GIFT round uses 33 cycles for 32 S-box look-ups and one pipeline latency. As a result, The 40-round operation of GIFT requires $33 \times 40 = 1,320$ cycles.

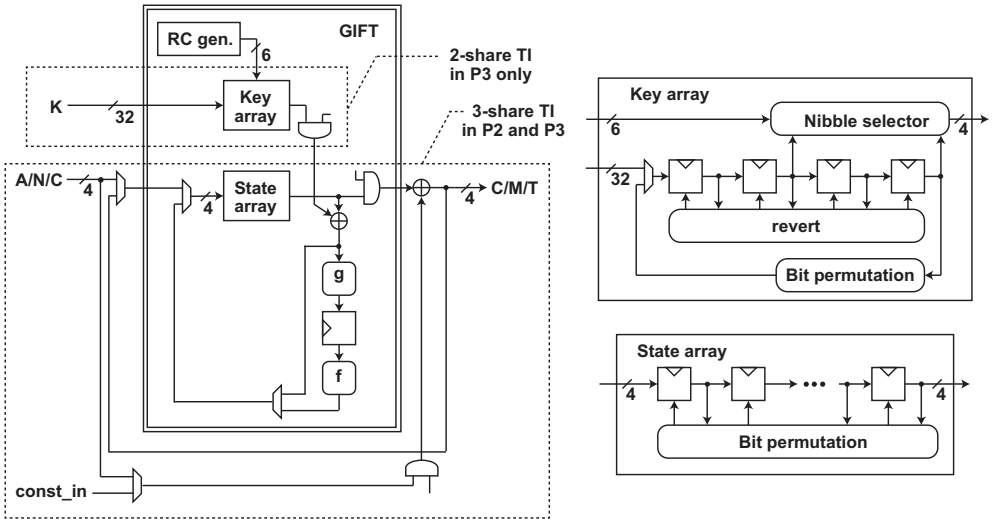


**Fig. 6.** SAEB[GIFT] hardware architecture.

### 5.3 Threshold Implementation

There is an option between protected and unprotected key/tweakey schedule. Conventional attacks such as differential power analysis (DPA) [18] cannot be used to attack key schedule that is independent of an attacker-controllable input e.g., plaintext or ciphertext. That is not generally true for TBCs, but SKINNY has the same property as far as the attacker-controllable tweak is placed in **TK3**, which is scheduled independently of **TK1** and **TK2**. Consequently, some previous works prioritize circuit area and use unprotected key-schedule implementations [6, 32, 37]. Meanwhile, if we consider a profiling attack on key/tweakey schedule, it is also reasonable to choose a protected key-schedule implementation. Considering the cost-security trade-off, we implement PFB[SKINNY] and SAEB[GIFT] with three different profiles: (**P1**) the unprotected implementation, (**P2**) TI with the unprotected key schedule and (**P3**) TI with the protected key schedule.

Table 2 summarizes the number of registers needed for the SKINNY and GIFT implementations for the different profiles. In (**P1**), both SKINNY and GIFT use 256 bits in total. In (**P2**), on the other hand, SKINNY use the smaller number of registers, 384 bits compared with 512 bits, because of the smaller block length. SKINNY still has a better performance in (**P3**) because key/tweakey schedule can be shared more efficiently. Since both GIFT and SKINNY have linear key/tweakey schedules, they can be realized with only two shares. Moreover, there is no need for protecting **TK3** of SKINNY that stores a public tweak. As a result, SKINNY and GIFT use 512 and 684 bits in (**P3**), respectively.

We use the formulae for the 3-share uniform S-boxes for SKINNY and GIFT from the conventional works [6] and [13], respectively. TI is implemented by duplicating the state/key/tweakey arrays and replacing the decomposed S-boxes ($f$ and $g$) with their shared maps. Fig. 5 and 6 show the boundaries of sharing for each profile.

**Table 2.** The number of registers for implementing SKINNY and GIFT in different profiles.

| Target | Profile | TI/State | TI/Key | State | Tweak/key | Total |
|---|---|---|---|---|---|---|
| SKINNY | (**P1**) | — | — | 64 | 192 | 256 |
| GIFT | (**P1**) | — | — | 128 | 128 | 256 |
| SKINNY | (**P2**) | ✓ | — | 192 | 192 | 384 |
| GIFT | (**P2**) | ✓ | — | 384 | 128 | 512 |
| SKINNY | (**P3**) | ✓ | ✓ | 192 | 320 | 512 |
| GIFT | (**P3**) | ✓ | ✓ | 384 | 256 | 640 |

**Table 3.** Breakdown of the post-synthesis circuit area of PFB[SKINNY] and SAEB[GIFT].

| Target | Component | Circuit area [GE] | | |
|---|---|---|---|---|
| | | (**P1**) | (**P2**) | (**P3**) |
| PFB[SKINNY] | Total | 3,111 | 4,492 | 5,858 |
| | Total/SKINNY | 2,956 | 4,284 | 5,649 |
| | Total/SKINNY/State array | 532 | 1,757 | 1,757 |
| | Total/SKINNY/Tweakey array | 2,062 | 2,062 | 3,419 |
| SAEB[GIFT] | Total | 2,761 | 5,037 | 6,229 |
| | Total/GIFT | 2,541 | 4,756 | 5,947 |
| | Total/GIFT/State array | 975 | 2,925 | 2,925 |
| | Total/GIFT/Key array | 1201 | 1,226 | 2,410 |

### 5.4 Performance Evaluation and Comparison

The designs are synthesized with the NanGate 45-nm standard cell library [27] using Synopsys Design Compiler while preserving the module hierarchy. Table 3 shows the breakdown of the post-synthesis performances.

We first discuss the unprotected implementations (**P1**). The circuit area of PFB[SKINNY] and SAEB[GIFT] are $3,111$ and $2,761$ [GE], respectively. SKINNY and GIFT dominate the circuit area of PFB[SKINNY] and SAEB[GIFT], The additional costs for the mode of operations are limited. The sizes of the state and key arrays are almost proportional to their register sizes, e.g., the 64-bit SKINNY state array (532 [GE]) is almost a half the size of the 128-bit GIFT state array (975 [GE]).

Although the PFB[SKINNY] implementation is larger than that of SAEB[GIFT] by 350 [GE], this is a positive result because (i) GIFT is known to have a better performance compared with SKINNY [6] and (ii) lightweight TBC is an emerging technology compared with lightweight block cipher. It is also note that PFB[SKINNY] is twice as fast as that of SAEB[GIFT]: PFB[SKINNY] and SAEB[GIFT] consume a 64-bit message/ciphertext block using 640 and $1,320$ cycles, respectively. Moreover, PFB has parallelizable encryption as discussed in Sect. 3.

Table 4 shows performance comparison with previous implementations. The unprotected implementations of SAEB[GIFT] and PFB[SKINNY] are smaller than previous implementations of AES-based AEs (SAEB[AES128] [25], CLOC[AES128], SILC[AES128], OTR[AES128] [3]). The bit-serial Ascon implementation without an interface has a smaller circuit area of $2,570$ [GE] [11];

however, the implementation needs an additional 128-bit key register to run another encryption/decryption with the same key. If we add the size of the key register (640 [GE] for 5 [GE/bit]) to 2,570 [GE], the Ascon implementation has the similar circuit size compared with that of PFB[SKINNY]. We also note that the Ascon implementation with an interface including a 128-bit key register has 3,750 [GE].

**Table 4.** Performance comparison; latency is that of a single call of a primitive (block cipher, tweakable block cipher, or permutation).

| Target | TI | Area [GE] | Latency [cycles] | Standard-cell library | Ref. |
|---|---|---|---|---|---|
| PFB[SKINNY] (**P1**) | — | 3,111 | 641 | NanGate 45-nm | **Ours** |
| SAEB[GIFT] (**P1**) | — | 2,761 | 1,320 | NanGate 45-nm | **Ours** |
| SAEB[AES128] | — | 3,502 | 231 | NanGate 45-nm | [25] |
| CLOC[AES128] | — | 4,310 | 210 | STMicro. 90-nm | [3] |
| SILC[AES128] | — | 4,220 | 210 | STMicro. 90-nm | [3] |
| OTR[AES128] | — | 6,770 | 210 | STMicro. 90-nm | [3] |
| Ascon w/o IF | — | 2,570 | 3,072 | UMC 90-nm | [11] |
| Ascon w/ IF | — | 3,750 | 3,072 | UMC 90-nm | [11] |
| Deoxys (Round*) | — | 11,936 | 14 | UMC 180-nm | [17] |
| Ketje-JR | — | 5,447 | 16 | NanGate 45-nm | [1] |
| PFB[SKINNY] (**P2**) | ✓ | 4,492 | 641 | NanGate 45-nm | **Ours** |
| PFB[SKINNY] (**P3**) | ✓ | 5,858 | 641 | NanGate 45-nm | **Ours** |
| SAEB[GIFT] (**P2**) | ✓ | 5,037 | 1,320 | NanGate 45-nm | **Ours** |
| SAEB[GIFT] (**P3**) | ✓ | 6,229 | 1,320 | NanGate 45-nm | **Ours** |
| Ascon w/o IF | ✓ | 7,970 | 3,072 | UMC 90-nm | [11] |
| Ascon w/ IF | ✓ | 9,190 | 3,072 | UMC 90-nm | [11] |
| Ketje-JR | ✓ | 18,335 | 16 | NanGate 45-nm | [1] |

We then discuss the protected implementations. With (**P2**), the PFB[SKINNY] implementation uses 4,492 [GE] which is smaller than that of SAEB[GIFT] (5,037 [GE]). That is explained by the smaller number of registers summarized in Table 2. PFB[SKINNY] is still advantageous with (**P3**): the circuit areas of PFB[SKINNY] and SAEB[GIFT] are 5,858 and 6,229 [GE], respectively. The protected PFB implementations are smaller than that of Ascon [11]) and Ketje [1] in conventional works as shown in Table 4. That is also explained by the number of registers. The sponge-based AEs have a relatively large state (384 bits for Ascon and 200 bits for Ketje-JR) that should be protected with three shares.

In summary, the unprotected PFB[SKINNY] implementation is competitive against the unprotected SAEB[GIFT] implementations and other conventional implementations. The benefit of a small block length, enable by PFB, becomes even larger with TI in which the number of registers are multiplied as shown in Table 2. As a result, the protected PFB[SKINNY] implementation outperforms that of SAEB[GIFT], Ascon [11], and Ketje [1].

## References


[1] Arribas, V., Nikova, S., Rijmen, V.: Guards in Action: First-Order SCA Secure Implementations of Ketje Without Additional Randomness. In: DSD 2018. pp. 492–499. IEEE Computer Society (2018)

[2] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A Block Cipher for Low Energy. In: ASIACRYPT 2015. LNCS, vol. 9453, pp. 411–436. Springer (2015)

[3] Banik, S., Bogdanov, A., Minematsu, K.: Low-area hardware implementations of CLOC, SILC and AES-OTR. In: HOST 2016. pp. 71–74. IEEE Computer Society (2016)

[4] Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In: CHES 2017. LNCS, vol. 10529, pp. 321–345. Springer (2017)

[5] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. IACR Cryptology ePrint Archive 2013, 404 (2013), http://eprint.iacr.org/2013/404

[6] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In: CRYPTO 2016. LNCS, vol. 9815, pp. 123–153. Springer (2016)

[7] Bhargavan, K., Leurent, G.: On the Practical (In-)Security of 64-bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 456–467. ACM (2016)

[8] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer (2007)

[9] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In: ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer (2012)

[10] Chakraborti, A., Iwata, T., Minematsu, K., Nandi, M.: Blockcipher-Based Authenticated Encryption: How Small Can We Go? In: CHES 2017. LNCS, vol. 10529, pp. 277–298. Springer (2017)

[11] Groß, H., Wenger, E., Dobraunig, C., Ehrenhöfer, C.: Suit up! - Made-to-Measure Hardware Implementations of ASCON. In: DSD 2015. pp. 645–652. IEEE Computer Society (2015)

[12] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer (2011)

[13] Gupta, N., Jati, A., Chattopadhyay, A., Sanadhya, S.K., Chang, D.: Threshold Implementations of GIFT: A Trade-off Analysis. IACR Cryptology ePrint Archive 2017, 1040 (2017), http://eprint.iacr.org/2017/1040

[14] Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: FSE 2006. LNCS, vol. 4047, pp. 310–327. Springer (2006)

[15] Iwata, T.: Authenticated Encryption Mode for Beyond the Birthday Bound Security. In: AFRICACRYPT 2008. LNCS, vol. 5023, pp. 125–142. Springer (2008)

[16] Iwata, T., Minematsu, K., Peyrin, T., Seurin, Y.: ZMAC: A Fast Tweakable Block Cipher Mode for Highly Secure Message Authentication. In: CRYPTO 2017. LNCS, vol. 10403, pp. 34–65. Springer (2017)

[17] Jean, J., Nikolic, I., Peyrin, T., Seurin., Y.: Deoxys v1.41. Submitted to the CAESAR competition (2016)

[18] Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: CRYPTO '99. LNCS, vol. 1666, pp. 388–397. Springer (1999)

[19] Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: FSE 2011. LNCS, vol. 6733, pp. 306–327. Springer (2011)

[20] Lampe, R., Seurin, Y.: Tweakable Blockciphers with Asymptotically Optimal Security. In: FSE 2013. LNCS, vol. 8424, pp. 133–151. Springer (2013)

[21] Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable Blockciphers with Beyond Birthday-Bound Security. In: CRYPTO 2012. LNCS, vol. 7417, pp. 14–30. Springer (2012)
[22] Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: EUROCRYPT 2014. LNCS, vol. 8441, pp. 275–292. Springer (2014)
[23] Minematsu, K., Iwata, T.: Tweak-Length Extension for Tweakable Blockciphers. In: IMACC 2015. LNCS, vol. 9496, pp. 77–93. Springer (2015)
[24] Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: A very compact and a threshold implementation of AES. In: Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings. pp. 69–88 (2011), https://doi.org/10.1007/978-3-642-20465-4_6
[25] Naito, Y., Matsui, M., Sugawara, T., Suzuki, D.: SAEB: A Lightweight Blockcipher-Based AEAD Mode of Operation. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2018(2), 192–217 (2018)
[26] Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. In: EUROCRYPT. LNCS, vol. 8441, pp. 257–274. Springer (2014)
[27] NanGate: NanGate FreePDK45 open cell library. http://www.nangate.com
[28] Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. In: ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer (2006)
[29] Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. J. Cryptology 24(2), 292–321 (2011)
[30] NIST: Submission requirements and evaluation criteria for thelightweight cryptography standardization process
[31] Peyrin, T., Seurin, Y.: Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In: CRYPTO 2016. LNCS, vol. 9814, pp. 33–63. Springer (2016)
[32] Poschmann, A., Moradi, A., Khoo, K., Lim, C., Wang, H., Ling, S.: Side-Channel Resistant Crypto for Less than 2, 300 GE. J. Cryptology 24(2), 322–345 (2011)
[33] Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006)
[34] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An Ultra-Lightweight Blockcipher. In: CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer (2011)
[35] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit Blockcipher CLEFIA (Extended Abstract). In: FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer (2007)
[36] Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE: A Lightweight Block Cipher for Multiple Platforms. In: SAC 2012. LNCS, vol. 7707, pp. 339–354. Springer (2013)
[37] Ueno, R., Homma, N., Aoki, T.: Toward More Efficient DPA-Resistant AES Hardware Architecture Based on Threshold Implementation. In: COSADE 2017. LNCS, vol. 10348, pp. 50–64. Springer (2017)

# Appendix

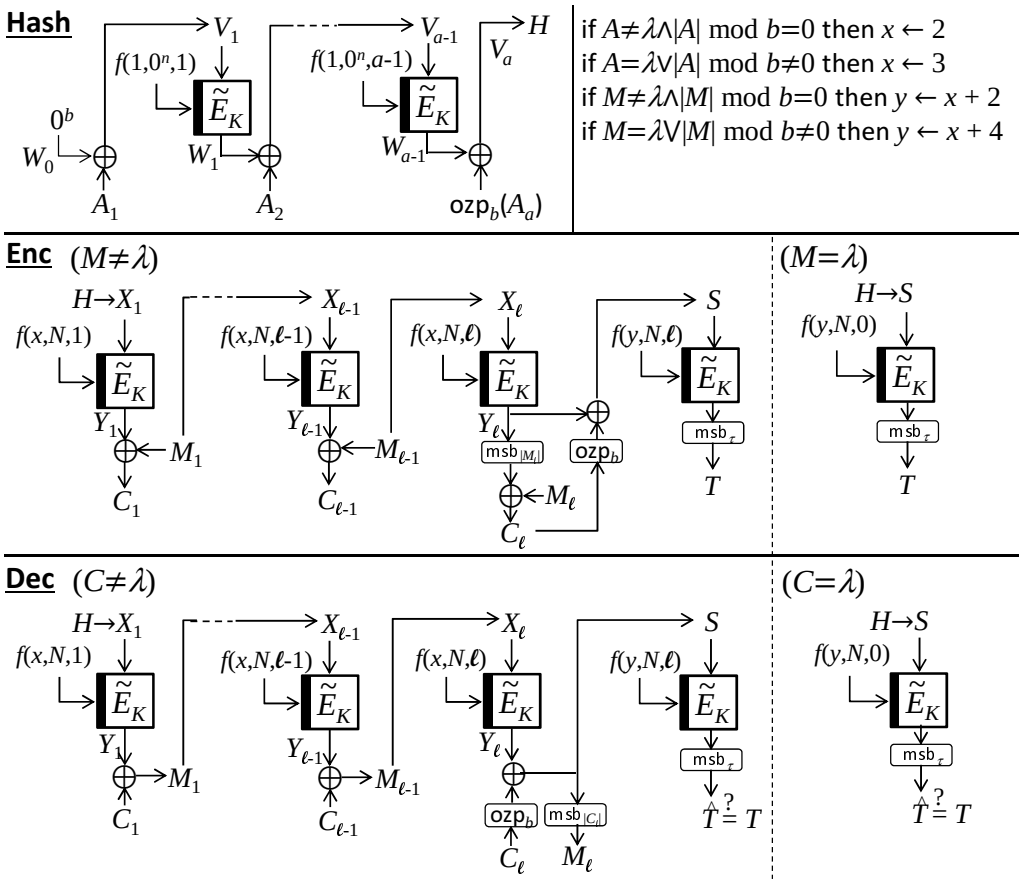


**Fig. 7.** PFB.