IACR Transactions on Symmetric Cryptology
ISSN 2519-173X, Vol. 2020, No. S1, pp. 417–445. DOI:10.13154/tosc.v2020.iS1.417-445

From Combined to Hybrid: Making Feedback-based AE even Smaller

Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Snehal Mitragotri, Mridul Nandi

Indian Statistical Institute, Kolkata, India
chakraborti.avik@lab.ntt.co.jp,nilanjan_isi_jrf@yahoo.com,ashwin.jha1991@gmail.com,snehal.mitr@gmail.com,mridul.nandi@gmail.com

Abstract. In CHES 2017, Chakraborti et al. proposed COFB, a rate-1 sequential block cipher-based authenticated encryption (AE) with only $1.5n$-bit state, where n denotes the block size. They used a novel approach, the so-called combined feedback, where each block cipher input has a combined effect of the previous block cipher output and the current plaintext block. In this paper, we first study the security of a general rate-1 feedback-based AE scheme in terms of its overall internal state size. For a large class of feedback functions, we show that the overlying AE scheme can be attacked in 2^{τ} queries if the internal state size is $n + \tau$ bits for some $\tau \geq 0$. This automatically shows that a birthday bound (i.e. $2^{n/2}$ queries) secure AE scheme must have at least $1.5n$-bit state, whence COFB is almost-optimal (use $1.5n$-bit state and provides security up to $2^{n/2}/n$ queries). We propose a new feedback function, called the *hybrid feedback* or HyFB, which is a hybrid composition of plaintext and ciphertext feedbacks. HyFB has a key advantage of lower XOR counts over the combined feedback function. This essentially helps in reducing the hardware footprint. Based on HyFB we propose a new AE scheme, called HyENA, that achieves the state size, rate, and security of COFB. In addition, HyENA has significantly lower XOR counts as compared to COFB, whence it is expected to have a smaller implementation as compared to COFB.

Keywords: COFB · feedback functions · authenticated encryption · lightweight · lower bound

1 Introduction

The era of the so-called Internet of Things (IoT)—communication networks interconnecting several small devices—is rapidly emerging. Security is one of the biggest concern in widespread adoption of IoT technologies. Authenticated encryption or AE is a symmetric-key cryptographic primitive that is expected to play a key role in securing IoT networks. This expectation is largely due to the fact that AE schemes can achieve both confidentiality and authenticity—two major concerns in information security. In particular lightweight[1] AE schemes have seen a sudden surge in demand. The recently concluded CAESAR competition and the ongoing NIST LwC project gave new impetus to the design and analysis of lightweight AE schemes. As a result, several lightweight AE schemes have mushroomed in recent years, including: Ascon [DEMS16], ACORN [Wu16], CLOC/SILC [IMG+,IMG+16], JAMBU [WH16] etc. from the CAESAR competition, and COFB [CIMN17a], Beetle [CDNY18], SUNDAE [BBLT18], SAEB [NMSS18] etc. from the ongoing NIST LwC project.

[1] Resource (memory, power etc.) constrained environments.

Received: 2019-06-01, Revised: 2019-09-01, Accepted: 2020-02-15, Published: 2020-06-22

1.1 Block Cipher-based Lightweight AE Schemes

Block ciphers have been the most popular choice for constructing AE schemes [KR16, Dwo11, BBLT18, ABD+16, CSR04, MP15, CIMN17a], though of late many permutation-based AE schemes [BJDAK16, CDNY18, BCDM, DHAK18] have also been proposed. In general, lightweight AE schemes follow sequential in nature. This is mainly due to two reasons. First, sequential modes have lesser hardware footprint. Second, lightweight environments rarely permit parallel computation infrastructure. In this work we study block cipher-based lightweight AE schemes with birthday bound security.

One of the most popular design paradigm of block cipher-based sequential AE schemes is as follows: First, the input to the AE scheme is parsed into n-bit blocks, where n denotes the block size of the underlying block cipher. Second, starting with a fixed initial state the parsed input is processed block-wise in sequential fashion. Third, a feedback function is applied on the previous block cipher output, some auxiliary secret state, and the current input (message or associated data) block to derive the next block cipher input, updated auxiliary secret state, and the current output (in case of message blocks). Thus, any block cipher based sequential AE scheme can be described by the underlying block cipher, the auxiliary secret state and the feedback function. Consequently, the efficiency and the hardware footprint of the overall AE scheme largely depends upon the underlying block cipher, the feedback function, and the auxiliary secret state.

In the following discussion, we assume that we have an ultra-lightweight and efficient block cipher to instantiate the AE scheme. Then the efficiency will depend upon the *Rate*—the number of data blocks processed per block cipher call—of the AE scheme and the *Latency*—time taken to produce the output—of the feedback function. Similarly, the circuit size of the feedback function and the internal state of the AE scheme will determine the overall hardware overhead of the AE schemes. A trivial upper bound on rate is 1. A trivial lower bound on the internal state of the AE scheme is $n + k$, where n and k denote the block and key size of the block cipher. A trivial lower bound on the circuit complexity of the feedback function is n-bit XORs[2].

An easy way to avoid the auxiliary secret state is to decrease the rate. Some rate-1/2 schemes like JAMBU [WH16], SAEB [NMSS18] and SUNDAE [BBLT18] follow this approach (we call it *approach 1*), albeit with slightly different security goals. Yet another approach is to keep the scheme rate-1 and concentrate on minimizing the auxiliary state and the circuit complexity of the feedback function. iFEED [ZWSW14] and COFB [CIMN17a] follow this approach (we call it *approach 2*). This latter approach has two advantages. First, the energy consumption will be lower as the number of block cipher calls is smaller. Second, if the auxiliary state and the feedback functions are optimal then the throughput/area ratio will be high due to higher rate and comparable hardware area. In this work, we focus on approach 2, i.e., we concentrate on rate-1 schemes.

1.1.1 Rate-1 Feedback-based AE Schemes

In [ZWSW14], Zhang et al. proposed a plaintext feedback-based mode iFEED that achieves optimal rate (i.e. rate-1). However, it requires a large state size of $(3n + k)$ bits. CPFB by Montes et al. [MP15] is a notable scheme which reduces the state size to $(2n + k)$ bits, at the cost of reducing the rate to 3/4. In CHES 2017, Chakraborti et al. [CIMN17a] proposed the first feedback-based AE scheme that achieves rate-1 with an impressive state size of just $1.5n + k$ bits. The main feature of COFB is a novel feedback function, called combined feedback. In the same paper they also posed the following question

How small can we go?

[2]Since, a feedback function takes $2n$-bit input and produces n-bit output, we need an n-bit binary operation. As XOR is one of the most simple operation, we believe that the trivial lower bound of the feedback function is n-bit XORs.

Here small means the hardware area, which directly depends on the internal state size and the circuit complexity of the feedback function. Thus, the answer to the above question depends on the answers to two sub-questions:

1. *What is the smallest internal state size?*
2. *What is the minimum circuit complexity of the feedback function?*

1.2 Our Contributions

This paper investigates the above two questions. Consequently, our contributions are twofold:

- General Rate-1 Feedback-based AE Scheme: We study a generalized feedback-based rate-1 AE scheme (see section 3). First, we give a linear algebra-based formalization of the underlying feedback function and identify some necessary properties of the associated feedback matrix. Second, based on this formalization we show the following results
    1. Any rate-1 feedback-based AE mode without any auxiliary state is insecure (see Lemma 1). Note that this also validates the decrease in rate in approach 1 discussed in section 1.1 above.
    2. We show that AE modes based on CFB or OFB are insecure even in the presence of auxiliary states. For AE modes based on PFB we show that at least n-bit auxiliary state is necessary for security.
    3. For any rate-1 feedback-based AE mode with τ-bit auxiliary state there exist an adversary that can break the mode in $O(2^\tau)$ many queries (see Lemma 2). Specifically, for any $\tau < n/2$ the AE mode can be broken in less than $2^{n/2}$ queries, *i.e. at least $n/2$-bit auxiliary state is necessary for security up to $2^{n/2}$ queries.* Thus, our result implies that COFB [CIMN17a] is almost-optimal [3] in terms of auxiliary state size.
- HyFB Feedback Function and the HyENA mode of AE: Although COFB achieves the minimum internal state size, we observe that the use of the combined feedback requires $2.5n$-bit XORs, which could be improved further.
    1. HyFB Feedback Function: With an aim of reducing the XOR counts, we propose a new feedback function, called the *Hybrid feedback* or HyFB, which is a hybrid of PFB and CFB (see section 4.1). Along the way, we study several hybrids and give some intuitive arguments on the insecurity of other combinations, namely PFB+OFB and CFB+OFB.
    2. HyENA Mode: We propose a new birthday bound secure authenticated encryption mode called HyENA based on the HyFB feedback function (see section 4.2). HyENA, like COFB, uses an $n/2$-bit auxiliary secret state and achieves almost-optimal internal state size of $1.5n + k$. However, HyENA significantly improves over the XOR counts in COFB, reducing the XOR counts from $2.5n$-bit to $1.5n$-bit. In short, HyENA achieves $(n/2 - \log n)$-bit security using overall $1.5n$-bit state and only $1.5n$-bit XOR[4] operations.

To summarize, we answer question 1 of section 1.1.1 with a concrete lower bound, and show that question 2 is worth exploring (by proposing HyFB, a lighter feedback function as compared to COFB). A concrete bound for question 2 (and a construction that achieves that bound) will close this line of research.

[3]By almost-optimal, we mean a nominal security degradation of a factor of n.

[4]We exclude the XORs required for constant field multiplications with 2 or 3 (required for updating the secret state) as the secret state update function is identical in both HyENA and COFB.

1.2.1 Comparative Results for HyENA

We provide a brief design comparison between HyENA and few existing feedback-based AE modes (in Table 1). Among rate-1 schemes, HyENA and COFB achieve almost-optimal state size, and HyENA has significantly better XOR count as compared to COFB. Among rate < 1, schemes JAMBU and SUNDAE are the closest competitors for HyENA. Although SUNDAE has better state size and XOR counts than HyENA, the rate is quite low which might not be desirable in certain scenarios. In summary, HyENA seems to be the best choice among rate-1 candidates.

We would like to point out that we do not include any parallel AE modes such as iFEED [ZWSW14], OCB3 [KR16], and OTR [Min16], in this comparison as they generally tend to have higher state sizes and their overall motivation is different.

Table 1: Comparison between AE modes using an n-bit block cipher with k-bit keys.

Scheme	Rate	State Size	XOR Count	Security	Ref
HyENA	1	$1.5n + k$	$1.5n$	$(n/2 - \log n)$	This work
COFB	1	$1.5n + k$	$2.5n$	$(n/2 - \log n)$	[CIMN17a, CIMN17b]
JAMBU	1/2	$1.5n + k$	$2n$	$n/2$	[WH16]
SUNDAE	1/2	$n + k$	n	$n/2$	[BBLT18]
CLOC/SILC	1/2	$2n + k$	n	$n/2$	[IMG+16]

2 Preliminaries

We fix the block size to n bits. We write $\{0,1\}^*$ and $\{0,1\}^n$ to denote the set of all binary strings (including the empty string λ), and the set of all binary strings of length n, respectively. $|X|$ denotes the number of the bits in the string X. For any $X \in \{0,1\}^n$, $\lceil X \rceil$ and $\lfloor X \rfloor$ denote the $n/2$ most and least significant bits of X respectively. For all practical purposes, we use the big endian format for representing binary strings, i.e. the least significant bit is the right most bit. We use the notation $\oplus$ to denote binary addition. For two strings $A, B \in \{0,1\}^\star$, $A\|B$ to denotes the concatenation of A and B. We use the notation $(X_{\ell-1}, \ldots, X_0) \xleftarrow{n} X$ to denote parsing of the string X into ℓ blocks such that for $0 \leq i \leq \ell - 2$, $|X_i| = n$ and $1 \leq |X_{\ell-1}| \leq n$. The expression $\mathcal{E}?\ a : b$ evaluates to a if $\mathcal{E}$ holds and b otherwise. For any binary string X with $|X| \leq n$, we define the padding function Pad as

$$\mathsf{Pad}(X) = \begin{cases} X & \text{if } |X| \bmod n = 0 \\ 0^{n-|X|-1}\|1\|X & \text{otherwise.} \end{cases}$$

Let $\gamma = (\gamma[1], \ldots, \gamma[s])$ be a tuple of equal-length strings. We define $\mathsf{mColl}(\gamma) = m$ if there exist distinct $i_1, \ldots, i_m \in [1..s]$ such that $\gamma[i_1] = \cdots = \gamma[i_m]$ and m is the maximum of such integer. We say that $\{i_1, \ldots, i_m\}$ is an m-multi-collision set for γ. For $n, \tau, \kappa \in \mathbb{N}$, E-n/κ denotes a block cipher family E, parametrized by the block length n, and key length κ. For $K \in \{0,1\}^\kappa$, and $M \in \{0,1\}^n$, we use $E_K(M) := E(K, M)$ to denote invocation of the encryption function of E on input K, and M. We fix positive even integers n, κ, r and t to denote the *block size*, *key size*, *nonce size*, and *tag size*, respectively, in bits. Throughout this document, we fix $n = 128$, and $\kappa = 128$, $r = 96$, and $t = n$. We use the notation $\boldsymbol{I}_{n\times n}$ and $\mathbf{0}_{n\times n}$ to denote $n \times n$ identity and zero bit-matrices, respectively. We drop the dimension of the matrix whenever it is clear form the context.

2.1 Finite Field Arithmetic

The set $\{0,1\}^{n/2}$ can be viewed as the finite field $\mathbb{F}_{2^{n/2}}$ consisting of $2^{n/2}$ elements. We interchangeably think of an element $A \in \mathbb{F}_{2^{n/2}}$ in any of the following ways: (i) as an $n/2$-

bit string $a_{\frac{n}{2}-1} \dots a_1 a_0 \in \{0,1\}^{n/2}$; (ii) as a polynomial $A(x) = a_{\frac{n}{2}-1}x^{n/2-1} + a_{\frac{n}{2}-2}x^{\frac{n}{2}-2} + \cdots + a_1 x + a_0$ over the field $\mathbb{F}_2$; (iii) a non-negative integer $a < 2^{n/2}$; (iv) an abstract element in the field. Addition in $\mathbb{F}_{2^{n/2}}$ is just bitwise XOR of two $n/2$-bit strings, and hence denoted by $\oplus$. $P(x)$ denotes the primitive polynomial used to represent the field $\mathbb{F}_{2^{n/2}}$, and α denotes the primitive element in this representation. The multiplication of $A, B \in \mathbb{F}_{2^{n/2}}$ is defined as $A \odot B := A(x) \cdot B(x) \pmod{P(x)}$, i.e. polynomial multiplication modulo $P(x)$ in $\mathbb{F}_2$. For $\frac{n}{2} = 64$, we fix the primitive polynomial [CSR05, IK03]

$$P(x) = x^{64} + x^4 + x^3 + x + 1. \tag{1}$$

Then, α, the primitive element, is $2 \in \mathbb{F}_{2^{64}}$. It is well-known [Rog04, KR11] that multiplication of any field element with α is computationally efficient. For any $A \in \mathbb{F}_{2^{64}}$, we have

$$A \odot \alpha = \begin{cases} A \ll 1 & \text{if } a_{|A|-1} = 0, \\ (A \ll 1) \oplus 0^{59}11011 & \text{if } a_{|A|-1} = 1. \end{cases}$$

Clearly, we need one shift and one conditional XOR. We refer to this process of multiplying any element $A \in \mathbb{F}_{2^{64}}$ with α, as *α-multiplication.*

2.2 Authenticated Encryption and Security Definitions

An authenticated encryption (AE) is an integrated scheme that provides both privacy of a plaintext $M \in \{0,1\}^*$ and authenticity of M as well as associated data $A \in \{0,1\}^*$. Taking a nonce N (which is a value unique for each encryption) together with associated data A and plaintext M, the encryption function of AE, $\mathcal{E}_K$, produces a tagged-ciphertext (C, T) where $|C| = |M|$ and $|T| = t$. Typically, t is fixed and we assume $n = t$ throughout the paper. The corresponding decryption function, $\mathcal{D}_K$, takes (N, A, C, T) and returns a decrypted plaintext M when the verification on (N, A, C, T) is successful, otherwise returns the atomic error symbol denoted by $\perp$.

Privacy. Given an adversary $\mathcal{A}$, we define the *PRF-advantage* of $\mathcal{A}$ against $\mathcal{E}$ as $\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{E}}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{E}_K} = 1] - \Pr[\mathcal{A}^{\$} = 1]|$, where \$ returns a random string of the same length as the output length of $\mathcal{E}_K$, by assuming that the output length of $\mathcal{E}_K$ is uniquely determined by the query. The PRF-advantage of $\mathcal{E}$ is defined as

$$\mathbf{Adv}^{\mathrm{prf}}_{\mathcal{E}}(q, \sigma, t) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{prf}}_{\mathcal{E}}(\mathcal{A}),$$

where the maximum is taken over all adversaries running in time t and making q queries with the total number of blocks in all the queries being at most σ. If $\mathcal{E}_K$ is an encryption function of AE, we call it the *privacy advantage* and write as $\mathbf{Adv}^{\mathrm{priv}}_{\mathcal{E}}(q, \sigma, t)$, as the maximum of all nonce-respecting adversaries (that is, the adversary can arbitrarily choose nonces provided all nonce values in the encryption queries are distinct).

Authenticity. We say that an adversary $\mathcal{A}$ *forges* an AE scheme $(\mathcal{E}, \mathcal{D})$ if $\mathcal{A}$ is able to compute a tuple (N, A, C, T) satisfying $\mathcal{D}_K(N, A, C, T) \neq \perp$, without querying (N, A, M) for some M to $\mathcal{E}_K$ and receiving (C, T), i.e. (N, A, C, T) is a non-trivial forgery.

In general, a forger is nonce-respecting with respect to encryption queries, but can make q_f forging attempts without restriction on N in the decryption queries, that is, N can be repeated in the decryption queries and an encryption query and a decryption query can use the same N. The *forging advantage* for an adversary $\mathcal{A}$ is written as $\mathbf{Adv}^{\mathrm{auth}}_{\mathcal{E}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathcal{E}_K, \mathcal{D}_K} \text{ forges}]$, and we write

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathcal{E}}((q_e, q_f), (\sigma_e, \sigma_f), t) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathrm{auth}}_{\mathcal{E}}(\mathcal{A})$$

to denote the maximum forging advantage for all adversaries running in time t, making q_e encryption and q_f decryption queries with total number of queried blocks being at most σ_e and σ_f, respectively.

Unified Security Notion for AE. The privacy and authenticity advantages can be unified into a single security notion as introduced in [FFL12, RS06]. Let $\mathcal{A}$ be an adversary that only makes non-repeating queries to $\mathcal{D}_K$. Then, we define the AE-advantage of $\mathcal{A}$ against $\mathcal{E}$ as

$$\mathbf{Adv}_{\mathcal{E}}^{\mathrm{AE}}(\mathcal{A}) = |\Pr[\mathcal{A}^{\mathcal{E}_K,\mathcal{D}_K} = 1] - \Pr[\mathcal{A}^{\$,\perp} = 1]|\,,$$

where the $\perp$-oracle always returns $\perp$ and the \$-oracle is as the privacy advantage. We similarly define $\mathbf{Adv}_{\mathcal{E}}^{\mathrm{AE}}((q_e,q_f),(\sigma_e,\sigma_f),t) = \max_{\mathcal{A}} \mathbf{Adv}_{\mathcal{E}}^{\mathrm{AE}}(\mathcal{A})$, where the maximum is taken over all adversaries running in time t, making q_e encryption and q_f decryption queries with the total number of blocks being at most σ_e and σ_f, respectively.

Pseudo Random Permutation. We use a block cipher E as the underlying primitive, and we assume the security of E as a PRP (pseudorandom permutation). The *PRP-advantage* of a block cipher E is defined as $\mathbf{Adv}_{E}^{\mathrm{prp}}(\mathcal{A}) = |\Pr[\mathcal{A}^{E_K} = 1] - \Pr[\mathcal{A}^{\mathsf{P}} = 1]|$, where P is a random permutation uniformly distributed over all permutations over $\{0,1\}^n$. We write

$$\mathbf{Adv}_{E}^{\mathrm{prp}}(q,t) = \max_{\mathcal{A}} \mathbf{Adv}_{E}^{\mathrm{prp}}(\mathcal{A})\,,$$

where the maximum is taken over all adversaries running in time t and making q queries. Here, σ does not appear as each query has a fixed length.

2.3 Coefficients-H Technique

We outline the Coefficients-H technique developed by Patarin, which serves as a convenient tool for bounding the advantage (see [Pat91, Vau03]). We will use this technique (without giving a proof) to prove our main theorem. Consider two oracles $\mathcal{O}_0 = (\$,\perp)$ (the ideal oracle for the game) and $\mathcal{O}_1$ (real, i.e. our construction in the same game). Let $\mathcal{V}$ denote the set of all possible views an adversary can obtain. For any view $\tau \in \mathcal{V}$, we will denote the probability to realize the view as $\mathsf{ip}_{\mathsf{real}}(\tau)$ (or $\mathsf{ip}_{\mathsf{ideal}}(\tau)$) when it is interacting with the real (or ideal respectively) oracle. We call these *interpolation probabilities.* Without loss of generality, we assume that the adversary is deterministic and fixed. Then, the probability space for the interpolation probabilities is uniquely determined by the underlying oracle. As we deal with stateless oracles, these probabilities are independent of the order of query responses in the view. Suppose we have a set of views, $\mathcal{V}_{\mathrm{good}} \subseteq \mathcal{V}$, which we call *good* views, and the following conditions hold:

1. In the game involving the ideal oracle $\mathcal{O}_0$ (and the fixed adversary), the probability of getting a view in $\mathcal{V}_{\mathrm{good}}$ is at least $1 - \epsilon_1$.
2. For any view $\tau \in \mathcal{V}_{\mathrm{good}}$, we have $\mathsf{ip}_{\mathsf{real}}(\tau) \geq (1 - \epsilon_2) \cdot \mathsf{ip}_{\mathsf{ideal}}(\tau)$.

Then we have $|\Pr[\mathcal{A}^{\mathcal{O}_0} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_1} = 1]| \leq \epsilon_1 + \epsilon_2$. The proof can be found in [Vau03].

3 Feedback-based Rate-1 Authenticated Encryption

In this section, we provide a simple generic structure for rate-1 feedback-based authenticated encryption modes. We assume that the only non-linear component considered in the mode is the underlying block cipher. Along with the main block cipher input-output state of

n-bit, the mode uses a τ-bit auxiliary secret state,[5] which can be used as another source of randomness. This auxiliary secret state can be viewed as some (non-linear) function on N. At each step, a feedback function is applied on the input data block, previous block cipher output, and the auxiliary secret state to generate the output data (ciphertext) block, next block cipher input and an updated auxiliary secret state. In case of associated data processing, the feedback function generates the next block cipher input and updates the auxiliary secret state. The generic structure is depicted in Fig. 1. We use the notation ρ^i to denote the i^{th} feedback function during the processing of plaintext blocks. For the associated data, we use the notation γ^i.

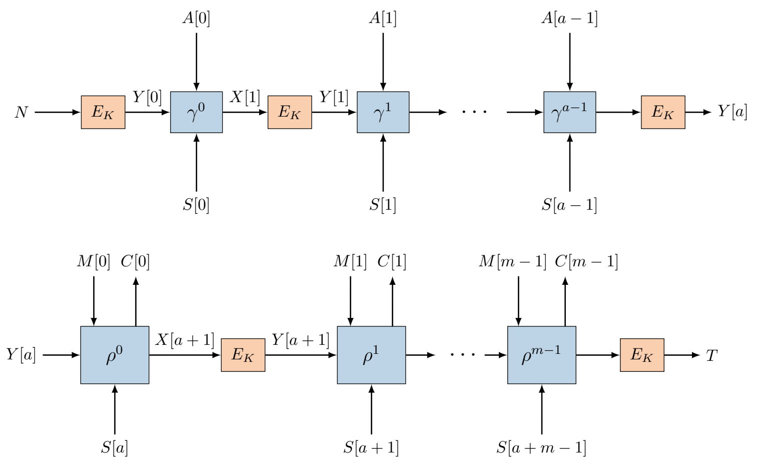


Figure 1: General Structure of Feedback-based Authenticated Encryption Modes.

3.1 Formal Representation of the Feedback Function

In this section, we provide a simple matrix based representation of the feedback function. Assuming the block size to be n and auxiliary secret state size to be τ, the linear feedback function ρ^i for the i^{th} block is defined by a $(2n+\tau) \times (2n+\tau)$ matrix E^i as given below:

$$\begin{pmatrix} X[i] \\ C[i] \\ S[i] \end{pmatrix} = E^i \cdot \begin{pmatrix} Y[i-1] \\ M[i] \\ S[i-1] \end{pmatrix},$$

where

$$E^i := \begin{pmatrix} (E^i_{y\to x})_{n\times n} & (E^i_{m\to x})_{n\times n} & (E^i_{s\to x})_{n\times\tau} \\ (E^i_{y\to c})_{n\times n} & (E^i_{m\to c})_{n\times n} & (E^i_{s\to c})_{n\times\tau} \\ (E^i_{y\to s})_{\tau\times n} & (E^i_{m\to s})_{\tau\times n} & (E^i_{s\to s})_{\tau\times\tau} \end{pmatrix}.$$

Here $(E^i_{v_1\to v_2})_{u\times v}$ denotes a submatrix of size $(u \times v)$ that shows the contribution of variable V_1 on V_2. Any rate-1 feedback-based authenticated encryption is defined by specifying these E^i matrices. The corresponding decryption function is given by

$$\begin{pmatrix} X[i] \\ M[i] \\ S[i] \end{pmatrix} = D^i \cdot \begin{pmatrix} Y[i-1] \\ C[i] \\ S[i-1] \end{pmatrix},$$

[5]The exact requirement of this secret state will be justified later.

where the decryption matrix is given by:

$$D^i = \begin{pmatrix} D^i_{y\to x} & D^i_{c\to x} & D^i_{s\to x} \\ D^i_{y\to m} & D^i_{c\to m} & D^i_{s\to m} \\ D^i_{y\to s} & D^i_{c\to s} & D^i_{s\to s} \end{pmatrix}.$$

The dimensions of decryption sub-matrices are analogous to the dimension of the encryption sub-matrices.

3.1.1 Necessary Properties of the Matrix

We now enumerate some necessary properties of the feedback matrices that will be used later in the section.

1. $E^i_{m\to c}$, $D^i_{c\to m}$ is invertible: This is due to the fact that $C[i]$ should have full effect of $M[i]$ (during encryption), and $M[i]$ should have full effect of $C[i]$ (during decryption). Otherwise, the correctness of the algorithm doesn't hold.

2. $[E^i_{y\to c} : E^i_{s\to c}]$ has rank n: If not, then there exists a $1 \times n$ row vector Δ such that $\Delta \cdot [E^i_{y\to c} : E^i_{s\to c}] = 0$. Thus, we can get a deterministic 1-bit relation $\Delta \cdot C = \Delta \cdot E^i_{m\to c} \cdot M$, which leads to a privacy attack.

3.1.2 Examples: Classical Feedback functions

We briefly mention some popular feedback functions:

1. Plaintext Feedback (PFB). Here the plaintext itself is used as the next block cipher input.

2. Ciphertext Feedback (CFB). Here the ciphertext is used as the next block cipher input.

3. Output Feedback (OFB). Here the previous block cipher output is used as the next block cipher input.

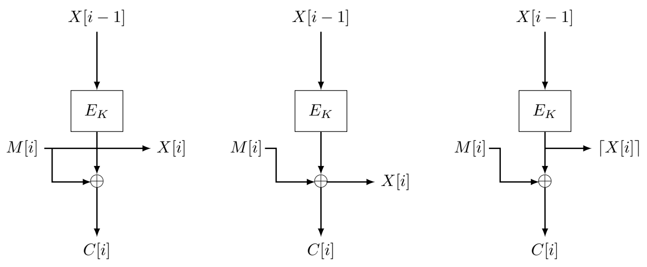


Figure 2: Types of Feedback: (i) PFB, (ii) CFB and (iii) OFB. This figure doesn't consider the effect of the auxiliary secret state

For all the classical feedback-based authenticated modes, we consider the traditional choice of (i) generating the ciphertext block by XORing the corresponding plaintext block with the previous block cipher output and (ii) updating the auxiliary secret state w/o depending on the plaintext or the previous block cipher output. Typically, the auxiliary secret state is updated by multiplying the previous state value with some constant field element (2 or 3) etc. Based on these traditional choices, we write down the encryption and decryption matrices for authenticated encryptions with PFB, CFB and OFB functions:

$$E^i_{\mathsf{OFB}} = \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix}, \quad E^i_{\mathsf{CFB}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix}, \quad E^i_{\mathsf{PFB}} = \begin{pmatrix} \mathbf{0} & \boldsymbol{I} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix},$$

Here $\star$ can be any binary matrix of appropriate dimension. The corresponding decryption matrices for these constructions are given as:

$$D^i_{\mathsf{OFB}} = \begin{pmatrix} \boldsymbol{I} & \mathbf{0} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix}, \quad D^i_{\mathsf{CFB}} = \begin{pmatrix} \mathbf{0} & \boldsymbol{I} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix}, \quad D^i_{\mathsf{PFB}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \star \\ \boldsymbol{I} & \boldsymbol{I} & \star \\ \mathbf{0} & \mathbf{0} & \star \end{pmatrix},$$

3.2 Insecurity of Rate-1 Feedback-based AE w/o Auxiliary State

Here we show the necessity of the secret auxiliary state for rate-1 feedback-based AE modes:

Lemma 1. *Any rate-*1 *feedback-based authenticated encryption mode without an auxiliary secret state is insecure.*

Proof. We describe an adversary $\mathcal{A}$ that makes 1 encryption query and then forges with a single decryption query.

1. $\mathcal{A}$ makes an authenticated encryption query $(N, A, M[0], M[1])$. Let the tagged ciphertext be $(C[0], C[1], T)$.

2. $\mathcal{A}$ computes the values of $Y[a]$ and $Y[a+1]$ from the following two equations:

$$C[0] = E^0_{y \to c} \cdot Y[a] + E^0_{m \to c} \cdot M[0], \quad C[1] = E^1_{y \to c} \cdot Y[a+1] + E^1_{m \to c} \cdot M[1].$$

   This follows from the fact that $\mathsf{rank}(E^i_{y \to c}) = n$, as it is invertible (see Sect. 3.1.1).

3. $\mathcal{A}$ calculates $C^f[0]$ such that

$$D^0_{y \to x} \cdot Y[a] \oplus D^0_{c \to x} \cdot C^f[0] = X[a+1].$$

   $\mathcal{A}$ can easily find such a $C^f[0]$ by computing at its own end.

4. Finally, $\mathcal{A}$ forges with $(N^f := N,\ A^f := A,\ C^f[0],\ T^f = T)$.

It is trivial to see that $\mathcal{A}$ forges with probability 1.

3.3 Rate-1 Feedback-based AE with Auxiliary State

In this section, we analyze the classical feedback-based AE as well as a newly proposed combined feedback-based AE with auxiliary state. The objective is to have an estimate on the size of the auxiliary state necessary to ensure birthday bound security (up to $2^{n/2}$ queries) for the AE construction.

3.3.1 Classical Feedback-based AE with Auxiliary State

We briefly discuss the security of classical feedback-based AE with auxiliary secret state. We first show that use of CFB or OFB doesn't provide a secure mode even with auxiliary secret state. Moreover, we claim that with PFB type feedback function, one needs n-bit auxiliary secret state to achieve the desired security.

1. Ciphertext Feedback-based (CFB) AE. Here an adversary exploits the fact that during decryption, previous block cipher outputs do not affect the next block cipher input: $D^i_{y\to x} = \mathbf{0}$, and hence an adversary can mount the following attack:

   - Make a query: $(N, A, M[1], M[2])$. Let ciphertext be $((C[1], C[2]), T)$.
   - Forge with $(N, A, (C'[1], C[2]), T)$, for any $C'[1] \neq C[1]$.

   Note that, the above attack would work even with auxiliary state.

2. Output Feedback-based (OFB) AE. Here an adversary exploits the fact that during decryption, ciphertext blocks do not affect the next block cipher input: $D^i_{c\to x} = 0$, and hence an adversary can mount the following attack:

   - Make a query: $(N, A, M[1])$. Let ciphertext be $(C[1], T)$.
   - Forge with $(N, A', C'[1], T)$, for any $(A', C'[1]) \neq (A, C[1])$.

   Similar to CFB, this mode also does not achieve security even with auxiliary secret state.

3. Plaintext Feedback-based (PFB) AE. Here an adversary can use the fact that the previous block cipher outputs do not affect the next block cipher input: $E^i_{x\to y} = 0$. It is easy to see that to achieve privacy (with query complexity $2^{n/2}$) of the mode, at least n-bit auxiliary state and it's full entropy is required.

In the above discussion, we see that PFB is the only choice for building an secure authenticated encryption, however at the cost of n-bit additional state. So, it was an interesting research direction whether it is possible to reduce the additional state size with some modified feedback functions.

3.3.2 Combined Feedback-based Authenticated Encryption

In CHES 2017, Chakraborti et al. [CIMN17a, CIMN17b] came up with a new feedback based function, termed *combined feedback* that uses a combination of previous block cipher output and the plaintext block to define the next block cipher input as depicted in Fig. 3.

The encryption and decryption matrix for this combined feedback function is given as:

$$E^i_{\mathsf{COFB}} = \begin{pmatrix} \boldsymbol{G} & \boldsymbol{I} & \boldsymbol{J}_{n/2} \\ \boldsymbol{I} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{U}^i \end{pmatrix}, \quad D^i_{\mathsf{COFB}} = \begin{pmatrix} \boldsymbol{G}+\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{J}_{n/2} \\ \boldsymbol{I} & \boldsymbol{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{U}^i \end{pmatrix},$$

where $\boldsymbol{G} \neq \boldsymbol{I}$ is an invertible matrix such that $\boldsymbol{I} + \boldsymbol{G}$ is also invertible (see [CIMN17a, CIMN17b] for the exact instance of $\boldsymbol{G}$), $\boldsymbol{U}_i$ matrix for COFB corresponds to the matrix for field mutiplication with $\mathbf{2}$ or $\mathbf{3}$ using the primitive polynomial $p(x) = x^{64} + x^4 + x^3 + x + 1$ over $\mathbb{F}_{2^{64}}$. We use the notation $\boldsymbol{J}_b$ to denote the matrix $\begin{pmatrix} \boldsymbol{I}_b \\ \mathbf{0}_b \end{pmatrix}$.
Based on this feedback they defined an authenticated encryption scheme called COFB that achieves the desired security with only $n/2$-bit additional masking state.

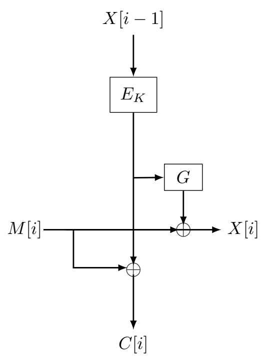


Figure 3: Combined Feedback

3.4 Optimality of the Auxiliary State Size

We show that $n/2$-bit additional state is necessary to achieve $2^{n/2}$ query security in any rate-1 feedback based authenticated encryption. More specifically, we prove the following lemma:

Lemma 2. *For any rate-*1 *feedback-based authenticated encryption mode with an additional state of size τ-bit, there exists an adversary that breaks the construction with query complexity of 2^τ-bits.*

Proof. Here we describe an adversary $\mathcal{A}$ that makes 1 encryption query and tries to forge with 2^τ many queries and succeeds in one forgery.

1. $\mathcal{A}$ makes an authenticated encryption query $(N, A, M = M[0]\|M[1])$. Let the tagged ciphertext be $(C[0]\|C[1]\|T)$.

2. Using the matrices E^1 and E^2, $\mathcal{A}$ obtains a matrix F such that the following holds:

$$F \cdot \begin{pmatrix} Y[a] \\ M[0] \\ S[a] \\ Y[a+1] \\ M[1] \end{pmatrix} = \begin{pmatrix} C[0] \\ C[1] \end{pmatrix}$$

   Note that, $S[a+1]$ can be written in terms of $Y[a], M[0]$ and $S[a]$, hence we do not consider $S[a+1]$ separately. As the values of $M[0]$ and $M[1]$ are known, one can represent the above as follows:

$$(F^1)_{2n\times(2n+\tau)} \cdot \begin{pmatrix} Y[a] \\ S[a] \\ Y[a+1] \end{pmatrix}_{(2n+\tau)\times 1} = \begin{pmatrix} C[0] \oplus \alpha \\ C[1] \oplus \beta \end{pmatrix}_{2n\times 1}$$

3. Here we claim that $\mathsf{rank}(F^1) = 2n$, as explained in the second necessary condition (see Sect. 3.1.1).

4. Since $\mathsf{rank}(F^1) = 2n$, using standard linear algebra results, $\mathcal{A}$ can find $(F^2)_{\tau\times(2n+\tau)}$

such that the following holds: $\begin{pmatrix} F^1 \\ F^2 \end{pmatrix}$ in invertible and

$$\begin{pmatrix} F^1 \\ F^2 \end{pmatrix} \cdot \begin{pmatrix} Y[a] \\ S[a] \\ Y[a+1] \end{pmatrix} = \begin{pmatrix} C[0] \oplus \alpha \\ C[1] \oplus \beta \\ \Delta \end{pmatrix}.$$

5. For each guess of Δ, the adversary $\mathcal{A}$ does the following:
   (a) $\mathcal{A}$ computes the values of $Y[a], S[a]$ and $Y[a+1]$:
   $$\begin{pmatrix} Y[a] \\ S[a] \\ Y[a+1] \end{pmatrix} = \begin{pmatrix} F^1 \\ F^2 \end{pmatrix}^{-1} \cdot \begin{pmatrix} C[0] \oplus \alpha \\ C[1] \oplus \beta \\ \Delta \end{pmatrix}.$$
   (b) $\mathcal{A}$ finds $C^f[0]$ such that
   $$D_{y \to x} \cdot Y[a] \oplus D_{c \to x} \cdot C^f[0] \oplus D_{s \to x} \cdot S[a] = X[a+1].$$
   (c) $\mathcal{A}$ tries to forge with $(N^f := N,\ A^f := A,\ C^f[0],\ T^f = T)$.

It is easy to see that by definition adversary $\mathcal{A}$ makes an attempt of exactly 2^τ many forging queries, and succeeds with probability 1 (as step 4 holds for at least 1 Δ value among all 2^τ values).

Remark 1. The previous attack can be easily extended to have a same length forgery attacks.

The above attack essentially shows that COFB indeed achieves the desired security with the optimal state size. However, COFB requires a total of $5n/2$-bit XORs (n-bit XORs to define the ciphertext, $3n/2$-bit XORs to define the next block cipher input that includes $n/2$-bit XORs for masking).

3.5 Hybrid Feedback

Motivated by the above question, in this section, we investigate on reducing the number of XORs further. For any construction it is quite intuitive that n-bit XORs are mandatory to define the ciphertext and the next block input. It also seems reasonable that one would require an additional $n/2$-bit XORs to incorporate the secret auxiliary state. Keeping this intuition in mind, we propose a new type of feedback function, called *hybrid* feedback, which requires $3n/2$-bit state as well as $3n/2$-bit XORs. As the name suggests, the hybrid feedback can use a combination of (PFB,CFB) or (PFB,OFB) or (CFB,OFB) to define the next block cipher input. Here we provide an informal intuitive argument on the security of AE modes with these hybrid feedback functions:

1. (PFB,CFB) Feedback: Since, this mode combines PFB and CFB, it achieves privacy up to $n/4$ bits (since encryption uses $n/2$ bits of CFB) but no authenticity without the secret auxiliary state. However, an additional $n/2$ bit auxiliary secret state (that is dependent on the nonce and updated with each block) can be XORed to the PFB part to provide the desired birthday bound security. Intuitively, this is due to the facts that (i) the additional state restricts state collision for different indices, and (ii) the use of (PFB,CFB) simultaneously in both encryption and decryption restricts state collisions at the same index.

2. (PFB,OFB) Feedback: As this mode combines PFB and OFB, and for the OFB part the updated state does not depend on the previous plaintext or ciphertext block, this hybrid feedback based mode will not be secure even with additional states. Here we briefly mention a simple attack on this mode:

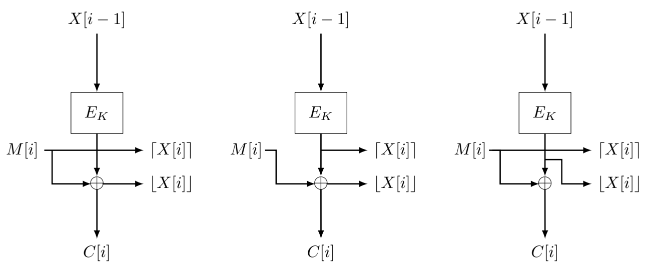


Figure 4: Types of Hybrid Feedback (from left to right): (PFB,CFB): $\lceil X[i] \rceil = \lceil M[i] \rceil$, $\lfloor X[i] \rfloor = \lfloor C[i] \rfloor$, (CFB,OFB): $\lceil X[i] \rceil = \lceil C[i] \rceil$, $\lfloor X[i] \rfloor = \lfloor Y[i] \rfloor$, (PFB,OFB): $\lceil X[i] \rceil = \lceil M[i] \rceil$, $\lfloor X[i] \rfloor = \lfloor Y[i] \rfloor$.

- Make a query: $(N, A, M[1], M[2])$. Let ciphertext be $(C[1], C[2], T)$.
- Forge with $(N, A, C[1], \lceil C[2] \rceil \| \star, T)$.

It is easy to see that this attack can be easily extended even if additional states are maintained.

3. (CFB,OFB) Feedback: This mode combines CFB and OFB, and the above attack holds in this case as well.

Based on the above discussion, from now on we write hybrid feedback or HyFB to denote (PFB,CFB) feedback, which can be represented by the following encryption matrix:

$$E^i_{\text{HyFB}} = \begin{pmatrix} \boldsymbol{J}_{n/2} & \boldsymbol{I} & \boldsymbol{K}_{n/2} \\ \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{U}_i \end{pmatrix}, \quad D^i_{\text{HyFB}} = \begin{pmatrix} \boldsymbol{K}_{n/2} & \boldsymbol{I} & \boldsymbol{K}_{n/2} \\ \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{U}^i \end{pmatrix},$$

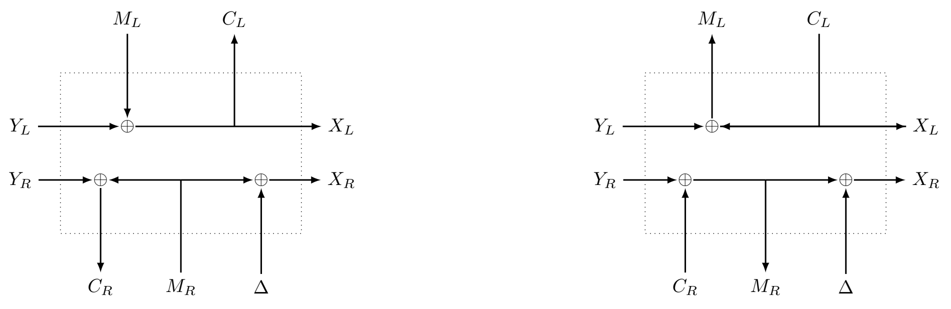
where the $\boldsymbol{U}^i$ matrix for HyFB is the same as COFB defined earlier and $\boldsymbol{K}_b$ matrix is defined as $\begin{pmatrix} \boldsymbol{0}_b \\ \boldsymbol{I}_b \end{pmatrix}$.

4 HyENA Authenticated Encryption Mode

In this section, we present the complete specification of HyENA. We also give detailed algorithmic descriptions for the modes. The HyENA authenticated encryption mode receives an encryption key $K \in \{0,1\}^\kappa$, a nonce $N \in \{0,1\}^r$, an associated data $A \in \{0,1\}^*$, and a message $M \in \{0,1\}^*$ as inputs, and returns a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^n$. The decryption algorithm receives a key $K \in \{0,1\}^\kappa$, an associated data $A \in \{0,1\}^*$, a nonce $N \in \{0,1\}^r$, a ciphertext $C \in \{0,1\}^*$ and a tag $T \in \{0,1\}^n$ as inputs and returns the plaintext $M \in \{0,1\}^{|C|}$, corresponding to the ciphertext C, if the tag T authenticates.

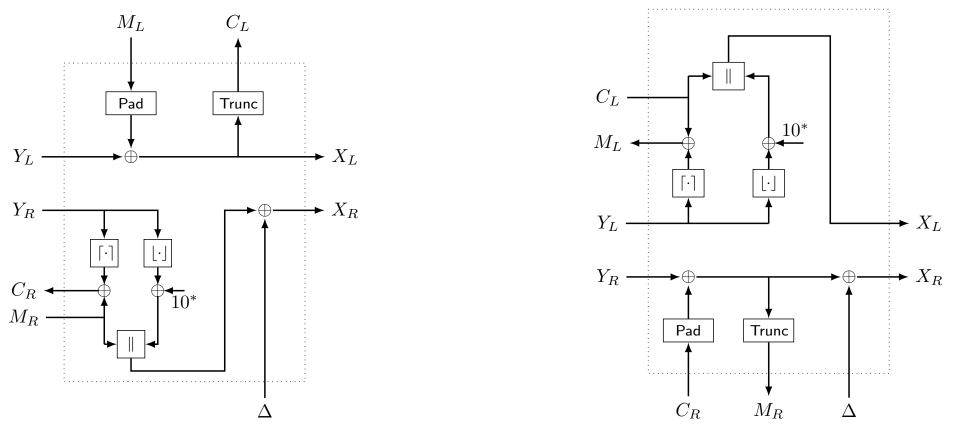
4.1 Hybrid FeedBack Function

The hybrid feedback function is illustrated in Figure 5 and 6. For any n-bit string S with $|S| \leq n$, we define S_L as the most significant $n/2$ bits of S, if $|S| \geq n/2$, otherwise S_L is set as S. Similarly, we define S_R as the least significant i bits of S, where $|S| = n/2 + i$. If $|S| \leq n/2$, S_R is empty.



(a) HYFB+ module. (b) HYFB- module.

Figure 5: HYFB module of HYENA for full data blocks. The number of XOR count is equals to $3n/2$.



(a) HYFB+ module. (b) HYFB- module.

Figure 6: HYFB module of HYENA for partial data blocks. Similar to the full block case, the number of XOR count here also equals to $3n/2$.

4.2 HYENA Mode

The complete specification of HYENA mode is presented in Algorithm 7. A pictorial description of HYENA is given in Figure 8.

4.2.1 Initialization

We define the initial state as

$$IV \leftarrow N\|0^{n-r-2}\|b_0\|b_1,$$

where b_0 is a bit indicating whether the associated data is empty or not ($b_0 = 0$ iff associated data is empty) and b_1 is a bit indicating whether there is any data or not

($b_1 = 0$ iff both associated data and plaintext are empty). Note that, the combination $(b_0, b_1) = (1, 0)$ is impossible. This initial vector is encrypted to generate the initial state $Y[0]$. The least significant $n/2$ bits of $Y[0]$ is considered as the masking value Δ. Formally, we define

$$\Delta \leftarrow \lfloor E_K(IV) \rfloor.$$

Remark 2. Here we would like to point out that, in the CAESAR API it is possible to get the values of b_0 and b_1 directly from the two signals EOT (End-Of-Type) and EOF (End-Of-File) resp. On the other hand, it is not possible to have an information on whether plaintext is empty or not at the beginning. By this initialization approach, we do not separate the cases of (non-empty AD, empty plaintext) and (non-empty AD, non-empty plaintext), rather we will take care of this separation in the finalization phase.

4.2.2 Associated Data Processing

For associated data processing, first we parse the associated data in n-bit blocks. We perform a 10^* padding on the associated data in the following cases: (i) when associated data is empty, and (ii) when final associated data block is partial. After the padding is done, we process the associated data blocks sequentially and update the state and the masking value as follows:

$$\begin{aligned} \Delta &\leftarrow 2 \cdot \Delta\,, \\ (X[i], \star) &\leftarrow \textsc{HyFB}+(Y[i-1], A[i], \Delta[i])\,, \\ Y[i] &\leftarrow E_K(X[i])\,, \end{aligned}$$

where $\star$ denotes some value that we do not bother about. To process the final associated data block, we multiply Δ by 3 (for full) or 3^2 (for partial) for the purpose of domain separation.

4.2.3 Plaintext Processing

For plaintext processing, first we parse the plaintext in n-bit blocks. We perform an 10^* padding on the plaintext only when the final plaintext block is partial. Note that, we do not perform any operation if plaintext is empty. After the padding is done, we process the plaintext blocks sequentially and updates the state and the masking value as follows:

$$\begin{aligned} \Delta &\leftarrow 2 \cdot \Delta\,, \\ (X[a+i], C[i]) &\leftarrow \textsc{HyFB}+(Y[a+i-1], M[i], \Delta[i])\,, \\ Y[a+i] &\leftarrow E_K(X[a+i]). \end{aligned}$$

To process the final plaintext block, we multiply Δ by 3 (for full) or 3^2 (for partial) for the purpose of domain separation.

4.2.4 Tag Generation

To generate the tag, we swap the most significant and least significant $n/2$ bits of the state and performs a block cipher encryption:

$$T \leftarrow E_K(X_R[a+m] \| X_L[a+m])$$

```
Algorithm HyENA-Enc$(K, N, A, M)$
 1. $Y \leftarrow$ Init$(N, A, M)$
 2. $(X, \Delta) \leftarrow$ Proc-AD$(Y, A)$
 3. if $|M| \neq 0$ then
 4.     $(X, C) \leftarrow$ Proc-TXT$(X, \Delta, M, +)$
 5. $T \leftarrow$ Tag-Gen$(X)$
 6. return $(C, T)$
```

```
Algorithm Init$(N, A, M)$
 1. $b_0 \leftarrow (|A| = 0)?\ 1 : 0$
 2. $b_1 \leftarrow (|A| + |M| = 0)?\ 1 : 0$
 3. $Y \leftarrow E_K(N \| 0^{n-r-2} \| b_1 \| b_0)$
 4. return $Y$
```

```
Algorithm Proc-AD$(Y, A)$
 1. $\Delta \leftarrow Y_R$
 2. if $|A| = 0$ then
 3.     $\Delta \leftarrow 3^2 \odot \Delta$
 4.     $(X, \star) \leftarrow$ HyFB+$(Y, \Delta, 0^{n-1}1)$
 5.     return $(X, \Delta)$
 6. else
 7.     $(A_{a-1}, \ldots, A_0) \overset{n}{\leftarrow} A$
 8.     for $i = 0$ to $a - 2$
 9.         $\Delta \leftarrow 2 \odot \Delta$
10.         $(X, \star) \leftarrow$ HyFB+$(Y, \Delta, A_i)$
11.         $Y \leftarrow E_K(X)$
12.     $t \leftarrow (|A_{a-1}| = n)?\ 1 : 2$
13.     $\Delta \leftarrow 3^t \odot \Delta$
14.     $(X, \star) \leftarrow$ HyFB+$(Y, \Delta, A_{a-1})$
15.     return $(X, \Delta)$
```

```
Algorithm Tag-Gen$(X)$
 1. $T \leftarrow E_K(X_R \| X_L)$
 2. return $T$
```

```
Algorithm HyENA-Dec$(K, N, A, C, T)$
 1. $Y \leftarrow$ Init$(N, A, M)$
 2. $(X, \Delta) \leftarrow$ Proc-AD$(Y, A)$
 3. if $|C| \neq 0$ then
 4.     $(X, M) \leftarrow$ Proc-TXT$(X, \Delta, C, -)$
 5. $T' \leftarrow$ Tag-Gen$(X)$
 6. if $T' = T$ then return $M$
 7. else return $\bot$
```

```
Algorithm HyFB+$(Y, \Delta, M)$
 1. $C \leftarrow \mathsf{Trunc}_{|M|}(Y) \oplus M$
 2. $\overline{M} \leftarrow \mathsf{Pad}(M)$, $\overline{C} \leftarrow \mathsf{Pad}(C)$
 3. $B \leftarrow \left(\overline{M}_L \| (\overline{C}_R \oplus \Delta)\right)$
 4. $X \leftarrow B \oplus Y$
 5. return $(X, C)$
```

```
Algorithm HyFB−$(Y, \Delta, C)$
 1. $M \leftarrow \mathsf{Trunc}_{|C|}(Y) \oplus C$
 2. $\overline{M} \leftarrow \mathsf{Pad}(M)$, $\overline{C} \leftarrow \mathsf{Pad}(C)$
 3. $B \leftarrow \left(\overline{M}_L \| (\overline{C}_R \oplus \Delta)\right)$
 4. $X \leftarrow B \oplus Y$
 5. return $(X, M)$
```

```
Algorithm Proc-TXT$(X, \Delta, D, \mathsf{dir})$
 1. $(D_{d-1}, \ldots, D_0) \overset{n}{\leftarrow} D$
 2. for $i = 0$ to $d - 2$
 3.     $\Delta \leftarrow 2 \odot \Delta$
 4.     $Y \leftarrow E_K(X)$
 5.     if $\mathsf{dir} = +$ then
 6.         $(X, O_i) \leftarrow$ HyFB+$(Y, \Delta, D_i)$
 7.     else
 8.         $(X, O_i) \leftarrow$ HyFB−$(Y, \Delta, D_i)$
 9. $t \leftarrow (|D_{d-1}| = n)?\ 1 : 2$
10. $\Delta \leftarrow 3^t \odot \Delta$
11. $Y \leftarrow E_K(X)$
12. if $\mathsf{dir} = +$ then
13.     $(X, O_{d-1}) \leftarrow$ HyFB+$(Y, \Delta, D_{d-1})$
14. else
15.     $(X, O_{d-1}) \leftarrow$ HyFB−$(Y, \Delta, D_{d-1})$
16. return $(X, (O_{d-1} \| \ldots \| O_0))$
```

Figure 7: Formal Specification of HyENA Authenticated Encryption and Decryption algorithm. We use the notation $\star$ to denote values that we do not care.

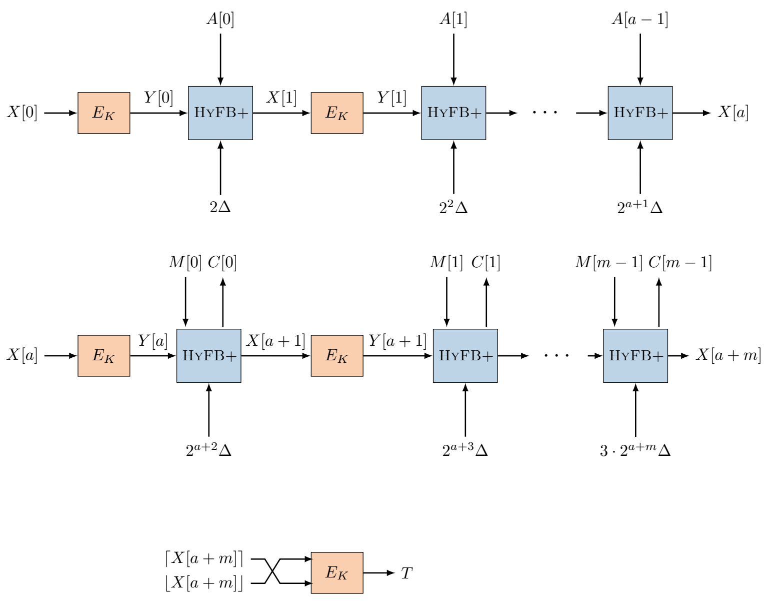


Figure 8: HyENA authenticated encryption mode for full data blocks.

4.3 Recommended Instantiation

In this section, we recommend an instance of HyENA. HyENA is parametrized by the choice of the underlying block cipher E and the nonce size r. We instantiate the block cipher with GIFT-128/128 [BPP+17] with 128-bit key and 128-bit block. Note that, the block cipher is well-defined and in this paper we do not include any description of GIFT-128/128. We fix $r = 96$.

4.4 Design Rationale

Here we briefly mention the rationale behind the design of HyENA.

4.4.1 Choice of Hybrid Feedback

To achieve a reduced memory implementation, it is natural to use sequential feedback at the cost of losing parallelizability. There exists several feedback based modes such as CFB, PFB and OFB (described earlier) and none of them fulfill our needs either in terms of the security bound or in terms of the storage size (all of them need to store an n-bit secret mask to achieve the birthday bound security) while processing data with rate 1. However, COFB fulfills our security requirements as it optimizes the storage size with the birthday bound security (it needs $n/2$-bit secret mask storage to achieve the birthday bound security), but COFB needs higher XOR counts. Hybrid feedback mode optimizes both the parameters as

- it provides birthday bound security with only $n/2$-bit additional storage (same as COFB) as well as
- it has much lower XOR counts as compared to COFB.

Considering all the above points, the hybrid feedback optimizes both the storage size as well as reduces the number of xor gates over all the existing feedback based modes with rate 1.

4.4.2 Choice of the Masking

We adopt a standard masking procedure equipped with an efficient domain separation technique. We would first like to mention that we maintain a secret mask Δ throughout the construction where Δ is computed by the $n/2$ least significant bits of the encrypted nonce. Note that, we can safely extract Δ during the associated data processing phase as it is not exposed (during message processing this values are exposed using the message and the ciphertext blocks). Thus, when the associated data is empty, we need an extra block cipher call to process a dummy empty block. From the implementation perspective, the masking function takes the current Δ and updates it by multiplying with either 2, 3 or 3^2 depending on the domains. Mathematically, we can represent this function as $\mathsf{mask} : \{0,1\}^{n/2} \times \mathbb{N}^2 \to \{0,1\}^{n/2}$ as

$$\mathsf{mask}(\Delta, a, b) = 2^a \odot 3^b \odot \Delta,$$

where we efficiently control a and b to separate the domains. The paper [Rog04] showed that $\forall (a,b) \in \{0, \cdots, 2^{51}\} \times \{0, \cdots, 2^{10}\}$, $2^a \odot 3^b$ are distinct for the specific primitive polynomial we use in this construction. This masking is similar to the one used in COFB.

5 Security

In this section, we provide the security of HyENA, mainly we prove the following Theorem:

Theorem 1.

$$\begin{aligned}\mathbf{Adv}^{\mathrm{AE}}_{HyENA}(q_e, q_v, \sigma_e, \sigma_v, t) \leq &\mathbf{Adv}^{\mathrm{prp}}_{E_K}(q', t') + \frac{2\sigma_e}{2^{n/2}} + \frac{\sigma_e^2}{2^n} + \frac{max\{n, nq_e/2^{n/4}\}}{2^{n/4}} \\ &+ \frac{nq_e}{2^{n/2}} + \frac{max\{n, nq_e/2^{n/4}\}q_e}{2^{3n/4}} + \frac{3nq_v}{2^{n/2}} \\ &+ \frac{2max\{n, nq_e/2^{n/4}\}q_v}{2^{3n/4}} + \frac{nq_v}{2^{3n/4}} + \frac{q_v}{2^n} + \frac{2n\sigma_v}{2^{n/2}}.\end{aligned}$$

where $q' = q_e + \sigma_e + q_v + \sigma_v$ which corresponds to the total number of block cipher calls through the game and $t' = t + O(q')$.

Proof. Without loss of generality, we can assume that $q' \leq 2^{n/2-1}$, since otherwise the right hand side becomes more than 1. The first transition we make is to use an n-bit (uniform) random permutation P instead of E_k and then use an n-bit (uniform) random function R instead of P. The probabilities corresponding to these two transitions are given in the first two terms of our bound. They are derived from the standard PRP-PRF switching lemma and from the computation of information theoretic reduction. The rest of the proof is done in the following sections.

5.1 Notations and Set-up

Fix a deterministic non-repeating query making distinguisher adv that interacts with either (1) the real oracle or (2) the ideal oracle making at most

1. q_e encryption queries $(N_i^+, A_i^+, M_i^+)_{i=1..q_e}$ with an aggregate of total σ_e many blocks and

2. attempts to forge with q_v many queries $(N_i^-, A_i^-, C_i^-, T_i^-)_{i=1..q_v}$ having a total of σ_v many blocks.

We assume that $\forall i$, M_i^+ and A_i^+ have m_i^+ and a_i^+ blocks respectively and C_i^- and A_i^- have c_i^- and a_i^- blocks respectively. We use the notation X, Y to denote the intermediate variables. Let

$$(S_i[0], S_i[1], \cdots, S_i[l_i^+ - 1]) \leftarrow (A_i^+[0], \cdots, A_i^+[a_i^+ - 1], M_i^+[0], \cdots, M_i^+[m_i^+ - 1])$$

where $l_i^+ = a_i^+ + m_i^+$. If bitwise representation of n-bit string G is $(G_{n-1} \cdots G_0)$. Then for $u > w$, we denote $(G_u \cdots G_w)$ by G_{u-w} which is a $u - w + 1$-bit substring of G from u^{th} bit to w^{th} bit of G.

5.2 Overview of the Attack Transcript

We begin with a description of the ideal oracle which consists of two phases.

- **Online phase**: For the i^{th} encryption query $(N_i^+, A_i^+ = (A_i[0], \ldots, A_i^+[a_i - 1]), M_i^+ = (M_i^+[0], \ldots, M_i^+[m_i - 1]))$, the oracle samples $(Y_i^+[a_i^+], \ldots, Y_i^+[l_i^+]) \leftarrow_\$ \{0,1\}^{n(m_i^+ + 1)}$ independently. It next sets the tag $T_i^+ = Y_i^+[l_i^+]$ and $C_i^+ = (C_i^+[0], \ldots, C_i^+[m_i^+ - 1])$ where $C_i^+[j] = Y_i^+[j + a_i^+] \oplus M_i^+[j]$ for $0 \le j \le m_i^+ - 1$ and returns (C_i^+, T_i^+) to $\mathcal{A}$.
- **Offline phase**: After $\mathcal{A}$ makes all the queries the oracle samples other Y^+ values as $Y_i^+[j] \leftarrow_\$ \{0,1\}^n$, for $0 \le j \le a_i - 1$.

For convenience, we slightly modify the experiment where we reveal to the adversary $\mathcal{A}$ (after $\mathcal{A}$ made all its queries and obtains corresponding responses but before it outputs its decision) the Y^+-values and now the adversary can set all intermediate values $X_i^+[j]$ using $S_i[j]$ and $Y_i^+[j]$. Note that $\Delta_i^+ = \lfloor Y_i^+[0] \rfloor$ and $\Delta_i^- = \lfloor Y_i^-[0] \rfloor$.
Overall, the transcript of the adversary $\tau := (\tau_e, \tau_v)$ be the list of queries and responses of $\mathcal{A}$ that constitutes the query response transcript of $\mathcal{A}$, where

- $\tau_e = (N_i^+, A_i^+, M_i^+, X_i^+, Y_i^+, T_i^+)_{i=1..q_e}$,
- $\tau_v = (N_j^-, A_j^-, C_j^-, T_j^-, \perp)_{j=1..q_v}$.

A prefix for a decryption query is defined as the common prefix blocks between the decryption query input string and an encryption query (if any) output string prepended with the nonce and the associated data. The length of the longest common prefix for the i^{th} decryption query is denoted as p_i. Note that if the decryption query uses a fresh nonce (not occurred during encryption queries), then it does not share any common prefix with any of the encryption queries then we set $p_i = -1$.

By ip_{ideal} and ip_{real} we denote the interpolation probability distribution of transcript τ induced by the ideal world and real world respectively. Note that, we use Patarin's Coefficient-H technique and below we start the proof by first identifying the bad events.

5.3 Identifying and Bounding Bad Events

Now, we define a set of events (initial bad events) for which the adversary aborts.

(i) B1: $\mathsf{mColl}(\Lambda) > n$ where Λ is the tuple of all $\lfloor X_i^+[l_i^+] \rfloor$ and $\lceil X_i^+[j] \rceil$ values for $0 \le j < l_i$.
This event signifies that n-multi-collision occurs in the upper part of the inputs of all blocks except the last blocks and the lower part of the inputs of all last blocks corresponding to the encryption queries.

(ii) B2: $\mathsf{mColl}(X^+[l]_{(n/2-1)-n/4}) > c$ where $c = \lceil nq_e/2^{n/4} \rceil$ and $X^+[l]_{(n/2-1)-n/4}$ is the tuple of all $X_i^+[l_i^+]_{(n/2-1)-n/4}$ values (it is the second $n/4$-bit chunk of $X_i^+[l_i^+]$. For example, when $n = 128$ it is $X_i^+[l_i^+]_{63-32}$) for $i \in [1 \cdots q_e]$,

(iii) B3: $X_i^+[j] = X_{i'}^+[0]$ for some $i, i' \in [1 \cdots q_e]$ and $0 < j < l_i^+$.

(iv) B4: $X_i^+[l_i^+] = X_{i'}^+[0]$ for some $i, i' \in [1 \cdots q_e]$

(v) B5: $X_i^+[j] = X_{i'}^+[j']$ for some $(i, j) \neq (i', j')$ and $j, j' > 0$.

(vi) B6: $X_i^-[p_i + 1] = X_{i'}^+[j']$ for some i, i' and $0 < j' < l_{i'}^+$.

(vii) B7: $X_i^-[p_i + 1] = X_{i'}^+[l_{i'}^+]$ for some i and i'.

(viii) B8: $X_i^-[p_i + 1] = X_{i_1}^+[0]$ and $X_i^-[p_i + 2] = X_{i'}^+[j']$ for some $p_i(\geq 0), i, i', i_1$ and $0 < j' \leq l_{i'}^+$.

The following lemma bounds the probability of bad transcripts in ideal oracle:

Lemma 3. *For any transcript τ,*

$$\begin{aligned} ip_{ideal}(\tau \in \mathcal{V}_{bad}) &\leq Pr[B1] + Pr[B2] + Pr[B3 \wedge B1^c] + Pr[B4 \wedge B2^c] + Pr[B5] \\ &+ Pr[B6 \wedge B1^c] + Pr[B7 \wedge B1^c \wedge B2^c] + Pr[B8 \wedge B1^c] \\ &\leq \frac{2\sigma_e}{2^{n/2}} + \frac{\sigma_e^2}{2^n} + \frac{max\{n, nq_e/2^{n/4}\}}{2^{n/4}} + \frac{nq_e}{2^{n/2}} + \frac{max\{n, nq_e/2^{n/4}\}q_e}{2^{3n/4}} + \frac{3nq_v}{2^{n/2}} \\ &+ \frac{2max\{n, nq_e/2^{n/4}\}q_v}{2^{3n/4}} + \frac{nq_v}{2^{3n/4}} \end{aligned}$$

Proof. Throughout the proof, we assume that all probabilities are defined over the ideal game. Here we provide the upper bounds for the bad events (in ideal oracle) one by one, as follows:

1. **Pr[B1].** The event B1 is a multi-collision event for randomly chosen n many $n/2$-bit strings out of σ_e many $n/2$-bit strings. As the Y^+-values are sampled uniformly and independently in the ideal game, we have,

$$Pr[B1] \leq \frac{\binom{\sigma_e}{n}}{2^{n/2(n-1)}} \leq \left(\frac{2\sigma_e}{2^{n/2}}\right)^n \leq \frac{2\sigma_e}{2^{n/2}}.$$

The last inequality follows from the assumption that $\sigma_e \leq 2^{n/2-1}$.

2. **Pr[B2]**.This event is a multicollision event for randomly chosen c ($\lceil nq_e/2^{n/4} \rceil$) many $n/4$-bit strings out of q_e many $n/4$-bit strings. Hence,

$$Pr[B2] \leq 2^{n/4}\left(\frac{\mathrm{e}q_e}{c2^{n/4}}\right)^c \leq 2^{n/4}\left(\frac{\mathrm{e}}{n}\right)^{\max\{n, nq_e/2^{n/4}\}} \leq 2^{n/4}\left(\frac{1}{2^n}\right) \leq \frac{1}{2^{3n/4}}.$$

The first inequality follows from the well-known results on multicollision [Gon81, BYG91] and e is the Euler's number. The third inequality follows from the assumption that $n \geq 2\mathrm{e}$.

3. **Pr[B3 $\wedge$ B1^c]**. Fix a pair of integers (i, j) such that $i \in [1 \cdots q_e]$. Then for $j < l_i^+$, we have

$$X_i^+[j] = N_{i'}^+ || 0^{n-r-2} || b_{1i'}^+ b_{0i'}^+$$

where r is the nonce size and we can express $\lfloor X_i^+[j] \rfloor$ as $\lfloor S_i[j] \rfloor \oplus 2^a \odot \Delta_i^+$ for some constant a. Here, Δ_i^+ is uniformly distributed. Note that there could be at most n many (i, j) indices for each i' and there are q_e many i' indices. Hence,

$$Pr[B3 \wedge B1^c] \leq \frac{nq_e}{2^{n/2}}.$$

4. **Pr[B4$\wedge$ B2^c]**. Fix an integer i such that $i \in [1 \cdots q_e]$. Then for $j = l_i^+$ and for some constants a, b, we have

$$\lceil N_{i'}^+ \rceil_{n/2} = \lfloor S_i[l_i^+ - 1] \rfloor \oplus 2^a \odot 3^b \odot \Delta_i^+ (= \lceil X_i^+[l_i] \rceil)$$

$$\lfloor N_{i'}^+ \rfloor_{n/4} || 0^{n-r-2} || b_{1i'}^+ b_{0i'}^+ = \lceil S_i[l_i^+ - 1] \rceil \oplus \lceil Y_i^+[l_i^+ - 1] \rceil (= \lfloor X_i^+[l_i] \rfloor).$$

Here, Δ_i^+ and $\lceil Y_i^+[l_i^+ - 1] \rceil$ are independent and uniformly distributed. Also i' can take at most q_e many values and B2^c implies that i can take at most c many values. Hence,

$$Pr[B4 \wedge B2^c] \leq \frac{cq_e}{2^{3n/4}} \leq \frac{max\{n, nq_e/2^{n/4}\}q_e}{2^{3n/4}}.$$

5. **Pr[B5].** For any $(i, j) \neq (i', j')$ and $j, j' > 0$, we have the following three possibilities: Case(i): $j < l_i^+, j' < l_{i'}^+$ Then for any $(i, j) \neq (i', j')$,the event $X_i^+[j] = X_{i'}^+[j']$ is nothing but two non-trivial linear equations. One is on $\lceil Y_i^+[j-1] \rceil$ & $\lceil Y_{i'}^+[j'-1] \rceil$ and other is on Const$_j \odot \Delta_i^+$ & Const$_{j'} \odot \Delta_{i'}^+$ for some constants Const$_j$ & Const$_{j'}$. For $i \neq i'$, we have $\lceil Y_i^+[j-1] \rceil, \lceil Y_{i'}^+[j'-1] \rceil, \Delta_i^+$ and $\Delta_{i'}^+$ are independent and uniformly distributed. For $i = i'$, we have Const$_j \neq$ Const$_{j'}$ and $\lceil Y_i^+[j-1] \rceil, \lceil Y_{i'}^+[j'-1] \rceil$ are independent and uniformly distributed. Hence this event has probability at most 2^{-n}. Therefore,

$$Pr[X_i^+[j] = X_{i'}^+[j']] \leq \frac{(\sigma_e - q_e)^2}{2^n}.$$

Similarly, we can argue for the other cases also.
Case(ii): For $j < l_i^+, j' = l_{i'}^+$ we have

$$Pr[X_i^+[j] = X_{i'}^+[j']] \leq \frac{(\sigma_e - q_e)q_e}{2^n}$$

Case(iii): For $j = l_i^+, j' = l_{i'}^+$, we have

$$Pr[X_i^+[j] = X_{i'}^+[j']] \leq \frac{q_e^2}{2^n}$$

Therefore,

$$Pr[B5] \leq \frac{(\sigma_e - q_e)\sigma_e + q_e^2}{2^n} \leq \frac{2\sigma_e^2}{2^n}.$$

6. **Pr[B6$\wedge$B1^c]** For $i \in [1 \cdots q_v]$ and $l_{i'}^+ > j' > 0$, we need to find the probability of the equality event $X_i^-[p_i + 1] = X_{i'}^+[j']$. The event B1^c implies that there are at most n many possible values for (i', j'). So if we fix (i', j') then we need to bound the probability for equality for the rest $n/2$ bits.
Case(i): $p_i = -1$. Here we have the equality event as $X_i^-[0] = X_{i'}^+[j']$. Since $p_i = -1$, $N_i^- \neq N_{i'}^+, \forall i' \in [1 \cdots q_e]$. Hence,

$$Pr[X_i^-[0] = X_{i'}^+[j'] \wedge B1^c] \leq \frac{nq_v}{2^{n/2}}.$$

Note that $\Delta_{i'}^+$ is uniformly distributed.
Case(ii): $0 \leq p_i < l_i^- - 1$. Since $p_i \geq 0$, we have $N_i^- = N_k^+$ for some k^{th} encryption query. Suppose $k \neq i'$. Then we obtain a non-trivial linear equation on $\Delta_{i'}^+$. Therefore, the probability in this case is at most $\frac{nq_v}{2^{n/2}}$.
Suppose $k = i'$. Then we must have $j' \neq p_i + 1$. Otherwise we get $C_i^-[p_i] = C_k^+[p_i]$

which contradicts the definition of p_i. Hence we get the probability at most $\frac{q_v}{2^{n/2}}$.
Case(iii): $p_i = l_i^- - 1$. Here we have

$$Pr[X_i^-[l_i^-] = X_{i'}^+[j'] \wedge B1^c] \leq \frac{nq_v}{2^{n/2}}.$$

In all of the above cases, we get the probability at most $\frac{nq_v}{2^{n/2}}$. Hence,

$$Pr[B6 \wedge B1^c] \leq \frac{nq_v}{2^{n/2}}.$$

7. **Pr[B7**$\wedge$ **B1**^{c}$\wedge$ **B2**^{c}].
Case(i): $p_i = -1$. Here, $\Delta_{i'}^+$ and $\lceil Y_{i'}^+[l_{i'}^+ - 1]\rceil$ are independent and uniformly distributed. Also i and i' can take at most q_v and c many values respectively. Then

$$Pr[X_i^-[0] = X_{i'}^+[l_{i'}^+] \wedge B2^c] \leq \frac{c\, q_v}{2^{3n/4}} \leq \frac{max\{n, nq_e/2^{n/4}\}q_v}{2^{3n/4}}.$$

Case(ii): $0 \leq p_i < l_i^- - 1$. Suppose $N_i^- \neq N_{i'}^+$. Then the equality event $X_i^-[p_i+1] = X_{i'}^+[l_{i'}^+]$ gives the two $n/2$-bit non-trivial equations, one is in Δ_i^- and another is in $\Delta_{i'}^+$. The event B1^c implies that i' can take at most n many values. Hence the probability of this event is at most $\frac{nq_v}{2^{n/2}}$.
Suppose $N_i^- = N_{i'}^+$. Then we must have $p_i + 1 \neq l_{i'}^+$ (otherwise we get contradiction to the definition of p_i).Therefore, the equality event $X_i^-[p_i+1] = X_{i'}^+[l_{i'}^+]$ gives a non-trivial equation in $\Delta_{i'}^+$. Then the probability is at most $\frac{q_v}{2^{n/2}}$.
Case(iii): $p_i = l_i^- - 1$. Here we get the same probability bound as in Case(ii). Hence,

$$Pr[B7 \wedge B1^c \wedge B2^c] \leq \frac{max\{n, nq_e/2^{n/4}\}q_v}{2^{3n/4}} + \frac{nq_v}{2^{n/2}}.$$

8. **Pr[B8**$\wedge$**B1**^{c}].Fix $i \in [1 \cdots q_v]$. Since $p_i \geq 0$, we have $N_i^- = N_k^+$ for some k^{th} encryption query. Then for a fixed i,

$$\begin{aligned} Pr[X_i^-[p_i+1] &= X_{i_1}^+[0]] = Pr[\lfloor Y_k^+[p_i]\rfloor \oplus \lfloor C_i^-[p_i]\rfloor \oplus Const_{p_i+1} \odot \Delta_k^+ \\ &= \lfloor N_{i_1}^+\rfloor_{n/4} || 0^{n-r-2} || b_{1i_1}^+ b_{0i_1}^+] \end{aligned}$$

Then we can bound this event with probability $1/2^{n/4}$.
Also the event B1^c implies that (i', j') can take at most n many values.

$$Pr[X_i^-[p_i+2] = X_{i'}^+[j'] \wedge B1^c] \leq \frac{n}{2^{n/2}}.$$

Hence,

$$Pr[B8 \wedge B1^c] \leq \frac{nq_v}{2^{3n/4}}.$$

By adding all these probabilities we prove the lemma. □

5.4 Lower Bound of $ip_{real}(\tau)$

We need to find out the lower bound for the ratio of ip_{real} and ip_{ideal}. For that, we fix $\tau \in \mathcal{V}_{good}$, where $\tau = (\tau_e, \tau_v)$ and

$$\tau_e = (N_i^+, A_i^+, M_i^+, X_i^+, Y_i^+, T_i^+)_{i=1..q_e},$$

$$\tau_v = (N_i^-, A_i^-, C_i^-, T_i^-, \perp)_{i=1..q_v}$$

. We assume that all the probability space (except for $ip_{ideal}(*)$) are defined over the real game. Clearly, $ip_{ideal}(\tau) = \frac{1}{2^{n(\sigma_e+q_e)}}$. Now we consider the real case. As **B3, B4, B5** do not hold for this good transcript τ, all the inputs of the random function inside τ_e are distinct and hence all the Y^+-values are independent and uniformly distributed. Also the X^+-values are uniquely determined from Y^+, A^+ and M^+.
Therefore, $\Pr[\tau_e] = \frac{1}{2^{n(\sigma_e+q_e)}}$. Now we have to calculate

$$\begin{aligned} ip_{real}(\tau) &= Pr[\tau_e, \tau_v] \\ &= Pr[\tau_v|\tau_e]Pr[\tau_e] \\ &= \frac{1}{2^{n(\sigma_e+q_e)}}Pr[\tau_v|\tau_e]. \end{aligned} \tag{2}$$

Let η be the event that $\forall i \in [1\cdots q_v]$, $X_i^-[j]$ for $p_i < j \leq l_i^-$ can not collide with any X^+-values in τ_e and $X_i^-[j]$'for $j \neq j'$. As the events **B6, B7, B8** can not hold for the good transcript, $Y_i^-[p_i+1]$ is uniformly random. Due to the property of feedback function, $X_i^-[p_i+2]$ is also uniformly random.
Now we need to calculate $Pr[\tau_v|\tau_e]$.

$$\begin{aligned} Pr[\tau_v|\tau_e] &= 1 - Pr[\tau_v^c|\tau_e] \\ &= 1 - (Pr[\tau_v^c, \eta|\tau_e] + Pr[\tau_v^c, \eta^c|\tau_e]) \end{aligned} \tag{3}$$

Here, $Pr[\tau_v^c, \eta|\tau_e]$ is the probability that $\exists i \in [1\cdots q_v]$ such that T_i^- is correct. But $T_i^- = Y_i^-[l_i^-]$ and the event η implies that $Y_i^-[l_i^-]$ is uniformly random. Hence $Pr[\tau_v^c, \eta|\tau_e]$ is the probability of guessing T_i^- correctly.
Therefore,

$$Pr[\tau_v^c, \eta|\tau_e] \leq \frac{q_v}{2^n} \tag{4}$$

Now, $Pr[\tau_v^c, \eta^c|\tau_e]$ and the event η^c can be described as
for $i \in [1\cdots q_v]$ and $p_i+1 \leq j \leq l_i^-$, $X_i^-[j] = X_{i_1}^+[j_1] \vee X_i^-[j] = X_i^-[j']$ for some $i_1, j_1, j' \neq j$ and $j' > p_i$.
The event B1^c can not hold for the good transcript. Hence (i_1, j_1) can take at most n many values. Then for a fixed i, we have
$Pr[X_i^-[j] = X_{i_1}^+[j_1]] \leq \frac{n.l_i^-}{2^{n/2}}$ and $Pr[X_i^-[j] = X_i^-[j'] \wedge T_i^-$ is correct$] \leq \frac{(l_i^-)^2}{2^{n/2}}\frac{1}{2^{n/2}}$. Also $\sum_{1\leq i\leq q_v}(l_i^-) \leq \sigma_v$ and $\sum_{1\leq i\leq q_v}(l_i^-)^2 \leq \sigma_v^2$.

Therefore,

$$Pr[\tau_v^c, \eta^c|\tau_e] \leq \frac{n\sigma_v}{2^{n/2}} + \frac{\sigma_v^2}{2^n} \leq \frac{2n\sigma_v}{2^{n/2}} \tag{5}$$

Combining (5), (6), (7) and (8), we get

$$\begin{aligned} ip_{real}[\tau] &\geq \frac{1}{2^{n(\sigma_e+q_e)}}\left(1 - \frac{q_v}{2^n} - \frac{2n\sigma_v}{2^{n/2}}\right) \\ &\geq ip_{ideal}[\tau]\left(1 - \frac{q_v}{2^n} - \frac{2n\sigma_v}{2^{n/2}}\right). \end{aligned}$$

The result follows from Coefficient-H technique combined with Lemma 1. □

6 Hardware Implementation of HyENA

HyENA aims to achieve a lightweight implementation on low resource devices. HyENA has a simple structure with a block cipher and a few linear operations. It has a small state

Table 2: Clock cycles per message byte for HyENA

	Message length (Bytes)											
	16	32	64	128	256	512	1024	2048	4096	16384	32768	262144
cpb	10.563	6.656	4.7031	3.727	3.238	2.9941	2.872	2.811	2.781	2.758	2.754	2.750

size and the complete circuit size is dominated by the underlying block cipher. In this section we provide hardware implementation details of HyENA instantiated with the GIFT block cipher.

6.1 Clock Cycle Analysis

We provide a conventional way for speed estimation, i.e, the number of clock cycles to process input bytes. Since HYENA processes at least one associated data (AD) block (one dummy block when AD is empty), we calculate the cpb assuming one AD block and m message blocks. We use 40 round GIFT and need 40 cycles for the GIFT module. We use 4 more cycles to compute the feedback and update the Δ value. Overall, HyENA needs $(44(m + 1) + 81)$ cycles. Table 2 shows the number of average cycles per input message bytes, which we call cycles per byte (cpb). The cpb is $(44(m + 1) + 81)/16m$ and it converges to 2.75 for very large m.

6.2 Hardware Architecture

This section describes the hardware implementation details of HyENA. The architecture is round based (without any pipelining) with 128-bit datapath and is modular. Due to the similarity between the associated data and the message processing phase, the same hardware module can be used for both. A single bit switching can be used to distinguish these data types. The hardware architecture is described in Fig. 9. The main architecture has the following modules.

1. **State Registers.** State registers store the block cipher intermediate states and the secret mask values after each clock cycles. There are one 128-bit **State** register to store the block cipher state, a 64-bit Δ register to store the Δ, and an 128-bit **Key** register to store the master key. **State** and **Key** are used by the *BC* module.

2. **BC Round.** GIFT_r module runs one GIFT round computation along with a round key update. The round key register is initialized with the master key from the **Key** register, each time the round function is invoked. The output from GIFT_r is sent to the **State** register, that is input to the next round. Each block process needs $r + 1$ cycles. After block process is done the value in **Key** is inverted to master key.

3. **Swap Module.** The Swap module just swaps the internal state by applying an $n/2$ bit left rotation.

4. **Update Module. Update** module updates the Δ register. It receives the current Δ value and updates it by multiplying with 2 or 3 or 3^2. Multiplication by 3^2 can be computed with 3 multiplication in two consecutive clock cycles. Overall we allocate total 4 clock cycles as we need maximum 4 cycle to compute $\alpha^a.(1 + \alpha)^b$ (i.e., $a + b \leq 4$). We use $3:1$ multiplexor implemented using scan registers.

Note that, we do not explicitly describe the control unit as we emphasize more on the explicit functional modules related to the pseudocode.

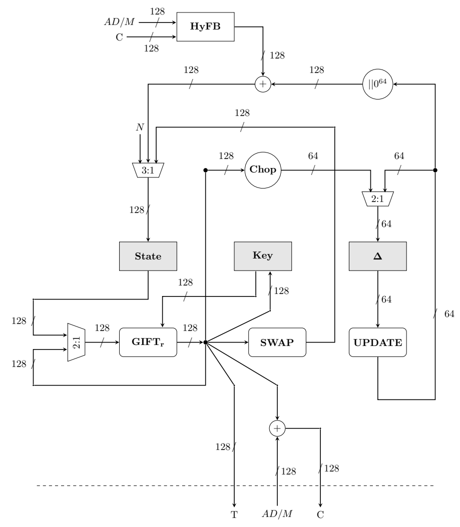


Figure 9: Hardware Architecture Diagram

Table 3: FPGA implementation results of HyENA

Design (Platform)	Slice Registers	LUTs	Slices	Frequency (MHz)	Throughput (Gbps)	Mbps/LUT	Mbps/Slice
HyENA (Virtex 7)	336	668	322	410.295	1.28	1.916	3.975

Basic Implementation. At the begining, the **State** register is loaded with N, next the initialization process encrypts N with GIFT. 64 bits from the encrypted nonce is truncated by the "chop" function. This chopped value is stored in Δ to initialize it. After the initialization, 128-bit AD blocks are fetched and sent to the HyFB module to compute an 128 bit intermediate state. It is next partially masked with the value in Δ for every GIFT call. After the AD is processed, the message blocks are processed in the same manner, except the ciphertext blocks are released. Finally, the tag is generated using an additional block cipher call.

6.3 Implementation Results

We implement HyENA on Xilinx Virtex 6 and Virtex 7, using VHDL and Xilinx ISE 13.4. Table 3 presents the implementation results of HyENA on Virtex 7.We follow the RTL approach and a basic iterative type architecture with 128-bit datapath. The ares are provided in the number of LUTs and slices. Frequency (MHz), Throughput (Gbps), and throughput-area efficiencies are also reported in addition to the hardware areas. Table 3 presents the mapped hardware results of HyENA.

The architecture for GIFT uses three registers $State$, RK and $Round$ to hold the blockcipher state, current round key and the round counter respectively. The architecture is divided into four modules SN, BP, ARK and ARC, $UKEY$. operations. SN module applies a 4-bit sbox to each of the 4-bit nibbles of the state. BP applies the bit permutation on the state. ARK performs the round key addition on the state and ARC applies round constant addition on the state. $UKEY$ updates the round key and stores it in RK. The architecture also uses another module EXT to extract a part of the round key to be added to the state.

Table 4: Comparison on Virtex 7 [ATH] (Results are taken by selecting **CAESAR Round 2** and **Standards** implemented results (following CAESAR API) in [ATH]). The results for CLOC-AES-Optimized and NORX-Optimized implementations (based on custom API, but not CAESAR API) have been taken from [KHKC17]. Results for COFB have been taken from [CIMN17b]. '-' implies data not available

Scheme	LUT	Slices	T'put (Gbps)	Mbps / LUT	Mbps / Slice
HyENA	668	322	1.28	1.916	3.975
COFB[AES]	1440	564	2.933	2.031	5.191
COFB[AES]-CAESAR-API	1496	579	2.747	1.842	4.395
COFB[GIFT]	771	316	2.230	2.892	6.623
COFB[GIFT]-CAESAR-API	1041	355	1.164	1.174	2.604
ACORN	499	155	3.437	6.888	22.174
AEGIS	7504	1983	94.208	12.554	47.508
AES-COPA	7795	2221	2.770	0.355	1.247
AES-GCM	3478	949	3.837	1.103	4.043
AES-OTR	4263	1204	3.187	0.748	2.647
AEZ	4686	1645	8.421	0.719	2.047
ASCON-128	1373	401	3.852	2.806	9.606
ASCON-128a	1836	506	5.476	2.982	10.821
CLOC-AES	3552	1087	3.252	0.478	1.561
CLOC-TWINE	1552	439	0.432	0.278	0.984
DEOXYS	3234	954	1.472	0.455	2.981
ELmD	4490	1306	4.025	0.896	3.082
JAMBU-AES	1595	457	1.824	1.144	3.991
JAMBU-SIMON	1200	419	0.368	0.307	0.878
Joltik	1261	390	0.402	0.319	1.031
Ketje-Jr	1567	518	4.080	2.604	7.876
Ketje-Sr	2592	724	6.752	2.605	9.326
Minalpher	2941	802	2.447	0.832	3.051
NORX	2881	857	10.328	3.585	12.051
PRIMATES-HANUMAN	1148	370	1.072	0.934	2.897
OCB	4269	1228	3.608	0.845	2.889
SCREAM	2315	696	1.100	0.475	1.580
SILC-AES	3040	910	4.365	1.436	4.796
SILC-LED	1682	524	0.267	0.159	0.510
SILC-PRESENT	1514	484	0.479	0.316	0.990
Tiaoxin	7556	1985	75.776	10.029	38.174
TriviA-ck	2221	684	14.852	6.687	21.713

6.4 Benchmarking with ATHENa Database

We benchmark our implemented results using the results in ATHENa Database [ATH], taking Virtex 7 (XC71140T) as our target platform. In Table 4, we provide comparisons on Virtex 7. Note that, we also add two custom API based implementation results (for CLOC-AES-Optimized and NORX-Optimized) provided in [KHKC17]. The other results in [KHKC17] have been collected in ASIC platform and hence are not compatible with our figures.

We also remark that it is basically hard to compare HyENA with other non-block-cipher-based designs in the right way, because of the difference in the primitives and the types of security guarantee. For example, ACORN is built from scratch and does not have any provable security result, and is subjected to several cryptanalyses [DRA16, SWB+16, SBD+16, LLMH16]. Sponge AE schemes (ASCON, Ketje, NORX, and PRIMATES-HANUMAN) use a keyless permutation of a large block size to avoid key scheduling circuit and have the provable security relying on the random permutation model.

Observation on comparison with COFB: Note that, HyENA is close to COFB in structure. However, the FPGA results show that HYENA achieves a fewer LUTs than GIFT-COFB but with a worse throughput. We would like to mention that, our main motivation is to optimize the hardware area rather than optimizing the throughput. Thus, overall our implementation performs very well in the number of LUTs at the cost throughput and throughput/area metric.

7 Conclusion

This paper completely characterizes feedback based rate 1 authenticated encryption modes and proposes a new feedback called *hybrid feedback* that combines plaintext and ciphertext feedback in a hybrid manner. Using this feedback, a new block cipher based mode for AE, called HYENA, is proposed that focuses on minimizing the state size and more importantly minimizing the XOR counts. When instantiated with an n-bit block cipher, HYENA operates at rate one with almost-optimal state size ($3n/2$ bits) as well as XOR count ($3n/2$ bits) and is provable secure up to $O(2^{n/2}/n)$ queries based on the standard PRP assumption on the block cipher.

References

[ABD+16] Elena Andreeva, Andrey Bogdanov, Nilanjan Datta, Atul Luykx, Bart Mennink, Mridul Nandi, Elmar Tischhauser, and Kan Yasuda. Colm v1. submission to caesar. 2016. https://competitions.cr.yp.to/round3/colmv1.pdf.

[ATH] Authenticated Encryption FPGA Ranking. https://cryptography.gmu.edu/athenadb/fpga_auth_cipher/rankings_view.

[BBLT18] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. Sundae: Small universal deterministic authenticated encryption for the internet of things. *IACR Transactions on Symmetric Cryptology*, 2018(3):1–35, Sep. 2018.

[BCDM] Tim Bayne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Elephant v1. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/elephant-spec.pdf.

[BJDAK16] Guido Bertoni, Michaël Peeters Joan Daemen, Gilles Van Assche, and Ronny Van Keer. Ketje v2. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/ketjev2.pdf.

[BPP+17] 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 Fischer and Homma [FH17], pages 321–345.

[BYG91] Ricardo A. Baeza-Yates and Gaston H. Gonnet. *Handbook of Algorithms and Data Structures in Pascal and C.* Addison-Wesley, 1991.

[CDNY18] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):218–241, 2018.

[CIMN17a] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In Fischer and Homma [FH17], pages 277–298.

[CIMN17b] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? *IACR Cryptology ePrint Archive*, 2017:649, 2017.

[CSR04] Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality . NIST Special Publication 800-38C, 2004. National Institute of Standards and Technology.

[CSR05] Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B, 2005. National Institute of Standards and Technology.

[DEMS16] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/asconv12.pdf.

[DHAK18] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of xoodoo and xoofff. *IACR Trans. Symmetric Cryptol.*, 2018(4):1–38, 2018.

[DRA16] Prakash Dey, Raghvendra Singh Rohit, and Avishek Adhikari. Full key recovery of ACORN with a single fault. *J. Inf. Sec. Appl.*, 29:57–64, 2016.

[Dwo11] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. NIST Special Publication 800-38D, 2011. csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.

[FFL12] Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In *FSE 2012*, pages 196–215, 2012.

[FH17] 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*. Springer, 2017.

[Gon81] Gaston H. Gonnet. Expected length of the longest probe sequence in hash code searching. *J. ACM*, 28(2):289–304, 1981.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In *FSE*, pages 129–153, 2003.

[IMG$^+$] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, and Eita Kobayashi. CAESAR Candidate CLOC. DIAC 2014.

[IMG$^+$16] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, and Eita Kobayashi. CLOC and SILC. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/clocsilcv3.pdf.

[KHKC17] Sachin Kumar, Jawad Haj-Yihia, Mustafa Khairallah, and Anupam Chattopadhyay. A comprehensive performance analysis of hardware implementations of CAESAR candidates. *IACR Cryptology ePrint Archive*, 2017:1261, 2017.

[KR11] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In *FSE*, pages 306–327, 2011.

[KR16] Ted Krovetz and Phillip Rogaway. OCB(v1.1). Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/ocbv11.pdf.

[LLMH16] Frédéric Lafitte, Liran Lerman, Olivier Markowitch, and Dirk Van Heule. SAT-based cryptanalysis of ACORN. *IACR Cryptology ePrint Archive*, 2016:521, 2016.

[Min16] Kazuhiko Minematsu. AES-OTR v3.1. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/aesotrv31.pdf.

[MP15] Miguel Montes and Daniel Penazzi. AES-CPFB v1. Submission to CAESAR. 2015. https://competitions.cr.yp.to/round1/aescpfbv1.pdf.

[NMSS18] Yusuke Naito, Mitsuru Matsui, Takeshi Sugawara, and Daisuke Suzuki. SAEB: A lightweight blockcipher-based AEAD mode of operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):192–217, 2018.

[Pat91] J. Patarin. Etude des Générateurs de Permutations Basés sur le Schéma du D.E.S. Phd Thesis de Doctorat de l'Université de Paris 6, 1991.

[Rog04] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*, volume 3329 of *Lecture Notes in Computer Science*, pages 16–31. Springer, 2004.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT*, pages 373–390, 2006.

[SBD+16] Md. Iftekhar Salam, Harry Bartlett, Ed Dawson, Josef Pieprzyk, Leonie Simpson, and Kenneth Koon-Ho Wong. Investigating cube attacks on the authenticated encryption stream cipher ACORN. In *ATIS 2016*, pages 15–26, 2016.

[SWB+16] Md. Iftekhar Salam, Kenneth Koon-Ho Wong, Harry Bartlett, Leonie Ruth Simpson, Ed Dawson, and Josef Pieprzyk. Finding state collisions in the authenticated encryption stream cipher ACORN. In *Proceedings of the Australasian Computer Science Week Multiconference*, page 36, 2016.

[Vau03] Serge Vaudenay. Decorrelation: A Theory for Block Cipher Security. *J. Cryptology*, 16(4):249–286, 2003.

[WH16] Hongjun Wu and Tao Huang. The JAMBU Lightweight Authentication Encryption Mode (v2.1). Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/jambuv21.pdf.

[Wu16] Hongjun Wu. ACORN: A Lightweight Authenticated Cipher (v3). Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/acornv3.pdf.

[ZWSW14] Liting Zhang, Wenling Wu, Han Sui, and Peng Wang. iFeed[AES] v1. Submission to CAESAR. 2014. https://competitions.cr.yp.to/round1/ifeedaesv1.pdf.