HyENA

Designers/Submitters:

Avik Chakraborti - NTT Secure Platform Laboratories, Japan
Nilanjan Datta - Indian Statistical Institute, Kolkata, India
Ashwin Jha - Indian Statistical Institute, Kolkata, India
Mridul Nandi - Indian Statistical Institute, Kolkata, India

avikchkrbrti@gmail.com, nilanjan_isi_jrf@yahoo.com, ashwin.jha1991@gmail.com,
mridul.nandi@gmail.com

March 29, 2019

Chapter 1

Introduction

In this document, we propose HyENA (HYbrid feedback-based ENcryption with Authentication) mode of operation that provides nonce-based authenticated encryption with associated data (NAEAD) functionality. Traditionally, block cipher based sequential encryption modes use one of the following methods, namely plaintext feedback (PFB), ciphertext feedback (CFB), or output feedback (OFB). HyENA is a hybrid feedback based mode (HyFB) as the block cipher input is partially ciphertext feedback and partially plaintext feedback.

HyENA primarily focuses on the hardware implementation cost. We aspire to minimize the state size overhead beyond the block cipher state (including the key schedule), and reduce the XOR counts. HyENA has several interesting features, most notably, it is single-pass (one block cipher call per data block), inverse-free (no need for block cipher decryption), and has very low state size approx. $1.5n + \kappa$ for block cipher with n-bit block and κ-bit key. All these features are quite desirable in NIST lightweight standardization process.

We instantiate HyENA with an ultra-lightweight block cipher GIFT-128 [1].

1.1 Notation

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 $X \in \{0,1\}^n$, X_L and X_R denote the $n/2$ most and least significant bits of X respectively. For all practical purposes, we use the little 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 $(X_{\ell-1}, \ldots, X_0) \xleftarrow{n} V$ to denote parsing of the string X into ℓ blocks such that for $0 \le i \le \ell - 2$, $|X_i| = n$ and $1 \le |X_{\ell-1}| \le n$. The expression $\mathcal{E}?\ a : b$ evaluates to a if $\mathcal{E}$ holds and b otherwise. For any binary string X with $|X| \le n$, we define the padding function Pad as

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

For any binary string X, the truncate function $\mathsf{Trunc}_i(X)$ returns the i least-significant bits of X.

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 $t = n$.

1.1.1 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 $a_{\frac{n}{2}-1} \ldots a_1 a_0 \in$

$\{0,1\}^{n/2}$; (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

$$P(x) = x^{64} + x^4 + x^3 + x + 1. \tag{1.1}$$

Then, α, the primitive element, is $2 \in \mathbb{F}_{2^{64}}$. It is well-known [5,6] 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*.

Chapter 2

Specification

In this chapter, we present the specification of HyENA along with its underlying block cipher GIFT-128. We also give detailed algorithmic descriptions for the modes. Finally, we give the concrete instantiation of HyENA with GIFT-128 block cipher.

2.1 HyENA Authenticated Encryption Mode

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 $T \in \{0,1\}^n$.

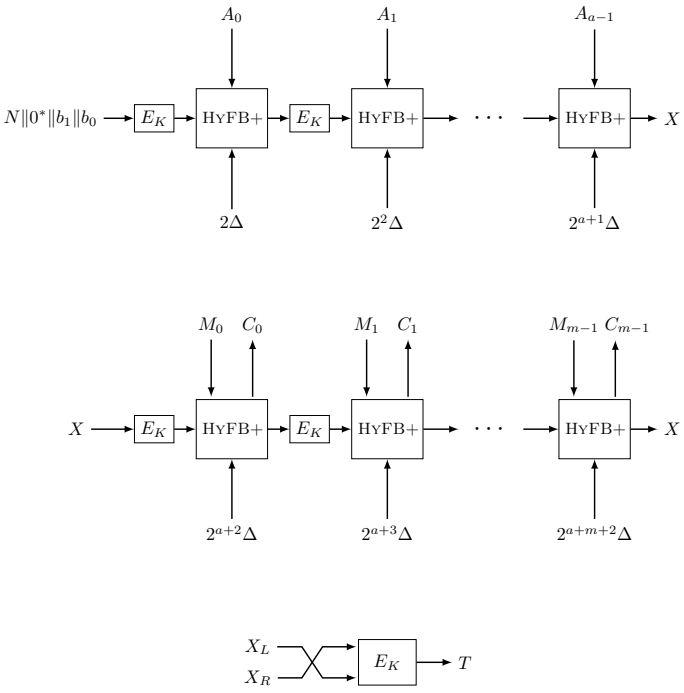
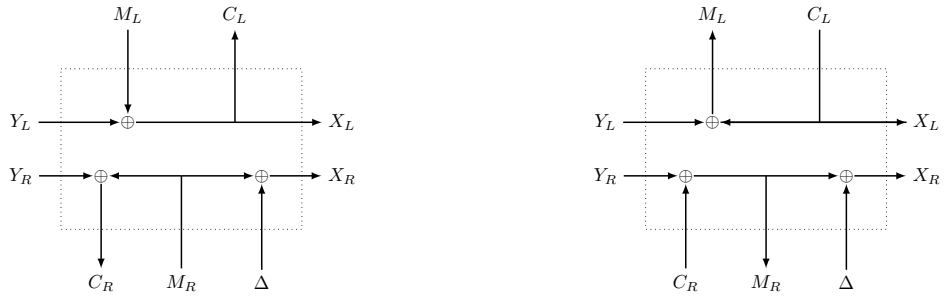


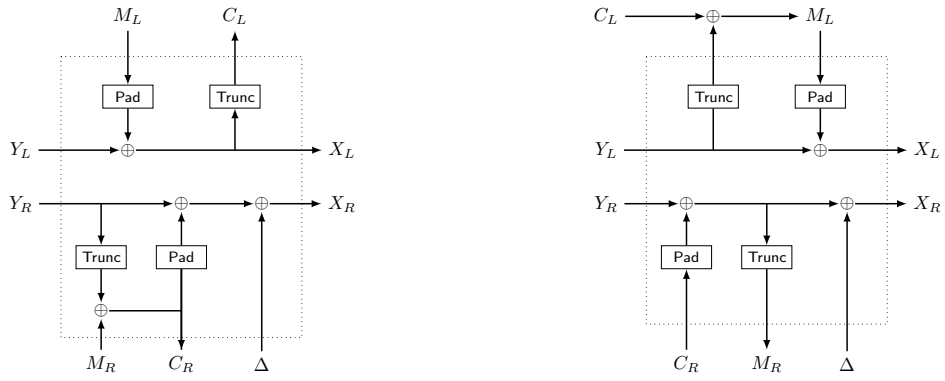
Figure 2.1: HyENA authenticated encryption mode for full data blocks.

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 return the plaintext $M \in \{0,1\}^{|C|}$, corresponding to the ciphertext C, if the tag T authenticates. Complete specification of HyENA is presented in Algorithm 2.4 and the corresponding pictorial description can be found in Figure 2.1. The feedback function is illustrated in Figure 2.2 and 2.3.



(a) HyFB+ module. **(b)** HyFB- module.

Figure 2.2: HyFB module of HyENA for full data blocks.



(a) HyFB+ module. **(b)** HyFB- module.

Figure 2.3: HyFB module of HyENA for partial data blocks.

Algorithm HyENA-Enc(K, N, A, M)

1. $Y \leftarrow \text{Init}(N, A, M)$
2. $(X, \Delta) \leftarrow \text{Proc-AD}(Y, A)$
3. **if** $|M| \neq 0$ **then**
4. $\quad (X, C) \leftarrow \text{Proc-TXT}(X, \Delta, M, +)$
5. $T \leftarrow \text{Tag-Gen}(X)$
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. $\Delta \leftarrow 2 \odot \Delta$
3. **if** $|A| = 0$ **then**
4. $\quad \Delta \leftarrow 2^2 \odot \Delta$
5. $\quad (X, \star) \leftarrow \text{HyFB+}(Y, \Delta, 0^{n-1}1)$
6. $\quad$ **return** (X, Δ)
7. **else**
8. $\quad (A_{a-1}, \ldots, A_0) \xleftarrow{n} A$
9. $\quad$ **for** $i = 0$ **to** $a - 2$
10. $\quad\quad (X, \star) \leftarrow \text{HyFB+}(Y, \Delta, A_i)$
11. $\quad\quad Y \leftarrow E_K(X)$
12. $\quad\quad \Delta \leftarrow 2 \odot \Delta$
13. $\quad t \leftarrow (|A_{a-1}| = n)?\ 1 : 2$
14. $\quad \Delta \leftarrow 2^t \odot \Delta$
15. $\quad (X, \star) \leftarrow \text{HyFB+}(Y, \Delta, A_{a-1})$
16. $\quad$ **return** (X, Δ)

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. $Y \leftarrow \text{Init}(N, A, M)$
2. $(X, \Delta) \leftarrow \text{Proc-AD}(Y, A)$
3. **if** $|C| \neq 0$ **then**
4. $\quad (X, M) \leftarrow \text{Proc-TXT}(X, \Delta, C, -)$
5. $T' \leftarrow \text{Tag-Gen}(X)$
6. **if** $T' = T$ **then return** M
7. **else return** $\perp$

Algorithm HyFB+(Y, Δ, M)

1. $C \leftarrow \mathsf{Trunc}_{|M|}(Y) \oplus M$
2. $\overline{M} \leftarrow \mathsf{Pad}(M)$, $\overline{C} \leftarrow \mathsf{Pad}(C)$
3. $B \leftarrow \big(\overline{M}_L \| (\overline{C}_R \oplus \Delta)\big)$
4. $X \leftarrow B \oplus Y$
5. **return** (X, C)

Algorithm HyFB−(Y, Δ, C)

1. $M \leftarrow \mathsf{Trunc}_{|C|}(Y) \oplus C$
2. $\overline{M} \leftarrow \mathsf{Pad}(M)$, $\overline{C} \leftarrow \mathsf{Pad}(C)$
3. $B \leftarrow \big(\overline{M}_L \| (\overline{C}_R \oplus \Delta)\big)$
4. $X \leftarrow B \oplus Y$
5. **return** (X, M)

Algorithm Proc-TXT$(X, \Delta, D, \mathsf{dir})$

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

Figure 2.4: Formal Specification of HyENA Authenticated Encryption and Decryption algorithm. For any n-bit string S, we define S_L (and S_R) as the most (and least) significant $n/2$ bits of S i.e. $(S_L, S_R) \xleftarrow{n/2} S$. We use the notation $\star$ to denote values that we do not care.

2.2 Specification of GIFT-128

GIFT-128 is a 128-bit block cipher proposed by [1]. GIFT-128 receives a 128-bit plaintext $b_{127}b_{126}\cdots b_0$ as the cipher state S where b_0 being the least significant bit. The cipher state can be viewed as 32 many

4-bit nibbles $S = w_{31}\|w_{30}\|\cdots\|w_0$. Along with the plaintext, the cipher also receives a 128-bit key $K = k_7\|k6\|\cdots\|k_0$ as the key state, where k_i is a 16-bit word. The cipher is composed of 40 rounds and each round is composed of the following operations:

SubCells: GIFT-128 uses an invertible 4-bit S-box and applies it to each nibble of the cipher state. Description of this S-box (in hex) is given in Table 2.1.

i	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Sbox(i)	1	A	4	C	6	F	3	9	2	D	B	7	5	0	8	E

Table 2.1: The GIFT-128 S-Box.

PermBits: To make the cipher light-weight, GIFT-128 employs an optimal bit permutation that satisfies a special property called "BOGI" [1]. The bit-permutation maps bit position i of the cipher state to bit position Perm(i), where

$$\text{Perm}(i) = 4\lfloor i/16 \rfloor + 32\Big(\big(3\lfloor (i \ \% \ 16)/4 \rfloor + (i \ \% \ 4)\big) \ \% \ 4\Big) + (i \ \% \ 4).$$

AddRoundKey: In this step a 64 bit round key is extracted from the key state, and the round key is xored with $\{b_{4i+1}\}$ and $\{b_{4i+2}\}$ of the cipher state. The round-keys are generated using a key scheduling algorithm which updates the key state at each round using some simple word-wise rotations and bit-wise rotations within a word.

AddRoundConstant: A single bit "1" and a 6 bit round constant are xored into the cipher state at bit position 127, 23, 19, 15, 11, 7 and 3 respectively. The round constants are generated using a 6 bit affine LFSR that was also used in SKINNY [2].

Detailed description of the algorithm can be found in [1].

2.3 Recommended Instantiation

We instantiate HyENA with GIFT-128 as the underlying block cipher, as our primary recommendation for AEAD. Here the key size is 128 bits, nonce size is 96 bits, and tag size is 128 bits.

Chapter 3

Security

In this chapter, we summarize the security details of HyENA. Section 3.1 gives the maximum data and time limits. It also lists all the relevant conditions to be adhered in order to maintain adequate security level. Section 3.2 presents a brief analysis against generic attacks (assuming the underlying block cipher is ideal), i.e. the security of modes. Section 3.3 presents a brief analysis on the security of GIFT-128, showing that it displays close to ideal behavior under the given data and time limit.

3.1 Security Claims

Table 3.1: Summary of security claims for HyENA. The data and time limits indicate the amount of data and time required to make the attack advantage close to 1.

Submissions	**Privacy**		**Integrity**	
	Time	Data (in bytes)	Time	Data (in bytes)
HyENA	2^{128}	2^{64}	2^{128}	2^{58}

In Table 3.1, we list the security levels of HyENA. We assume a nonce-respecting adversary, i.e. for a fixed key, no pair of distinct encryption queries share the same public nonce value, although we remark that the security may even hold when the public nonce value is sampled uniformly at random from the nonce space for each encryption query. All our security claims are based on full round GIFT-128, and we do not claim any security for HyENA with round-reduced variants of GIFT-128.

3.1.1 Statement

We declare that there are no hidden weaknesses in the HyENA mode of operation. Further, to the best of our knowledge, public third-party analysis do not raise any security threat to the submission HyENA, within the data and time limit prescribed in Table 3 of section 4.

3.2 Security Analysis of HyENA

We study the security of HyENA against generic attacks (assuming the underlying block cipher is ideal, i.e. random permutation). First we briefly explain possible attack strategies along with a rough lower bound estimate on the data and time complexity of each strategy. Then we validate the recommended criteria given in Table 3.1 by substituting concrete parameters. In the following discussion:

- D denotes the total (both encryption and decryption) data complexity. This parameter quantifies the online resource requirements, and includes the total number of blocks (among all messages and associated data) processed through the underlying block cipher for a fixed master key. We use D_e and D_v to account for the data complexity of encryption and decryption/verification queries.

- T denotes the time complexity. This parameter quantifies the offline resource requirements, and includes the total time required to process the offline evaluations of the underlying block cipher. Since one call of the block cipher can be assumed to take a constant amount of time, we generally take T as the total number of offline calls to the block cipher.

3.2.1 Privacy Security of HyENA

In privacy attacks the adversary is concerned with distinguishing the HyENA mode with an ideal authenticated encryption scheme, by exploiting access to the encryption algorithm. In other words, we are interested in the usual IND-CPA security notion. The adversary can distinguish the mode from ideal if there is no randomness in some ciphertext (or tag) blocks. This is possible in the following way:

- Encryption-Encryption Block Matching: For a pair of distinct encryption query blocks, the internal states matches. Then, the block that appears later will definitely have non-random behavior, though the adversary may not be able to detect it. In any case it is sufficient to bound the occurrence of this event. Now we may have two cases: (i) the two blocks belong to different query, in which case the nonce is different, and we can bound the probability of full state collisions, which is roughly $D_e^2/2^n$; (ii) the two blocks belong to the same query, in which case they must have different indices and hence we can again bound the probability of full state collision by at most $D_e^2/2^n$. The second case can be argued as follows: for the upper part of the state, we have uniform and random ciphertext block for the later query (contributing $\approx n/2$-bit entropy); and for the lower part the equation in Δ masking is again almost uniform and random (contributing $\approx n/2$-bit entropy). Note that the equation in Δ is non-trivial as the two colliding blocks have distinct indices. Combining the two cases we get $D_e \approx 2^{n/2}$. As we use the standard PRP notion on the underlying block cipher, we can safely assume $T \approx 2^\kappa$.

3.2.2 Integrity Security of HyENA

In this case the adversary has to forge a fresh and valid ciphertext and tag pair. The adversary is allowed to make encryption queries to the encryption algorithm and forging queries to the decryption algorithm. In other words, we use the INT-CTXT security notion.

In forgery attack, the adversary can apply one of the following strategies:

- Tag Guessing: The adversary can simply guess the tag in each decryption query. The probability of correct guess is roughly $D_v/2^n$. This gives $D_v \approx 2^n$.
- Decryption-Encryption Block Matching: Some decryption query block might match some encryption query block. Now depending upon the type of encryption query block we can have two cases:
  1. Non-init block: The encryption block is from AD or message processing. In this case we can first bound the probability of roughly n-multicollisions on the most significant part of any ciphertext blocks, which can be bounded by $D_v/2^{n/2}$. Given this we know that the decryption query has at most n many choices for the lower part. So a full state collision would occur with probability at most $nD_v/2^n$. This gives $D_v \approx 2^{n-\log_2 n}$.
  2. Init block: The decryption query block collides with the starting state $(N\|0^{n-r-2}\|b_1\|b_0)$ of some encryption query. In this case, one can show that the probability is bounded by at most $D_v/2^{n/2}$, which gives $D_v \approx 2^{n/2}$.

  Again using the standard PRP notion we can bound $T \approx 2^\kappa$.

3.2.3 Validation of Security Claims

The security claims given in Table 3.1 follow from the rough lower bounds on D_e, D_v, and T, as discussed in subsections 3.2.1 and 3.2.2, and the fact that $n = \kappa = 128$. Importantly, it can be observed that the HyENA mode is secure, as long as the upper limits on D_e, D_v, and T are not violated, and the underlying block cipher is a PRP.

3.3 Security Analysis of GIFT-128

We briefly mention the existing best possible analysis of GIFT-128. In [8], Zhu et al. uses the mixed-integer-linear-programming (MILP) based analysis to find an 18-round differential characteristic with probability 2^{-109}, which was further extended to mount a 23-round key recovery attack with time and data complexity of 2^{120} each and memory complexity of 2^{80}. Hence, we believe that 40 rounds of GIFT-128 should be secure against differential cryptanalysis.

GIFT-128 has a 9-round linear hull effect of $2^{-45.99}$, which implies that 27 rounds would achieve correlation potentially lower than 2^{-128}. Therefore, we expect 40 rounds of GIFT-128 has enough resistance against linear cryptanalysis.

The designers of GIFT-128 analysed the security of GIFT-128 against integral attacks. They have used the bit-based division property to the detect the longest integral distinguisher and found a 11-round integral distinguisher. No improved integral attacks have been reported since, and we believe full round of GIFT-128 has resistance against integral attacks.

The number of rounds of impossible differentials in GIFT-128 is much smaller than the integral attack, and 40 rounds are quite sufficient to resist the impossible differential attack.

Meet-in-the-middle attack exploits the property that a part of key does not appear during a certain number of rounds. The designers and the follow-up work by Sasaki [7] showed an attack against 15-rounds of GIFT-64 and mentioned the difficulty of applying it to GIFT-128 due to the larger ratio of the number of subkey bits to the entire key bits per round. Note that, in GIFT-128 one round uses 64 bits of the key and the entire key size is 128 bits.

Chapter 4

Features and Design Rationale

Here, we summarize the salient features and design rationale of HyENA:

1. **Inverse-Free**: HyENA is an inverse-free authenticated encryption algorithm. Both encryption and verified decryption of the algorithm do not require any decryption call to the underlying block cipher. This reduces the overall hardware footprint significantly, especially in the combined encryption-decryption implementations.

2. **Optimal**: HyENA requires $(a + m + 1)$ many block cipher invocations to process an a block associated-data and m block message. In [3], it has been shown that this is the optimal number of non-linear primitive calls required for any nonce based authenticated encryption. This feature is particularly important for short messages from the perspective of energy consumption, which is directly dependent upon the number of non-linear[1] primitive calls.

3. **Low State-size**: HyENA requires a state size as low as $3n/2$-bits along with the key state.

4. **Low XOR Count**: To achieve optimal, inverse-free authenticated ciphers with low state, a possible direction is to use the combined feedback approach where (i) the previous block cipher output is XORed with the plaintext to generate the ciphertext, and (ii) the next block cipher input is defined as the XOR of the plaintext with some linear function of the previous block cipher output. This technique was used in the popular authenticated encryption mode COFB [4]. It is easy to see that such combined feedback function require at least $2n$ bits of XOR operations (when operated on n bit data), along with some additional XOR operations required for the linear function mentioned above. On the contrary, in HyENA, we use the concept of hybrid feedback or HyFB, where the block cipher input is defined partially via ciphertext feedback and partially via plaintext feedback. This reduces the number of the XOR operations to only n bits.

[1] In general, non-linear operations consume significantly more energy as compared to linear operations

Bibliography

[1] 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 *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 321–345, 2017.

[2] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 123–153, 2016.

[3] Avik Chakraborti, Nilanjan Datta, and Mridul Nandi. On the optimality of non-linear computations for symmetric key primitives. *J. Mathematical Cryptology*, 12(4):241–259, 2018.

[4] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *CHES 2017*, pages 277–298, 2017.

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

[6] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *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*, pages 16–31, 2004.

[7] Yu Sasaki. Integer linear programming for three-subset meet-in-the-middle attacks: Application to gift. In Atsuo Inomata and Kan Yasuda, editors, *Advances in Information and Computer Security*, pages 227–243, Cham, 2018. Springer International Publishing.

[8] Baoyu Zhu, Xiaoyang Dong, and Hongbo Yu. Milp-based differential attack on round-reduced gift. Cryptology ePrint Archive, Report 2018/390, 2018. `https://eprint.iacr.org/2018/390`.