SAEB: A Lightweight Blockcipher-Based AEAD Mode of Operation

Yusuke Naito[1], Mitsuru Matsui[1], Takeshi Sugawara[2] and Daisuke Suzuki[1]

[1] Mitsubishi Electric Corporation, Japan
[2] The University of Electro-Communications, Japan

Abstract. Lightweight cryptography in computationally constrained devices is actively studied. In contrast to advances of lightweight blockcipher in the last decade, lightweight mode of operation is seemingly not so mature, yet it has large impact in performance. Therefore, there is a great demand for lightweight mode of operation, especially that for authenticated encryption with associated data (AEAD). Among many known properties of conventional modes of operation, the following four properties are essential for constrained devices:

1. **Minimum State Size:** the state size equals to a block size of a blockcipher.
2. **Inverse Free:** no need for a blockcipher decryption.
3. **XOR Only:** only XOR is needed in addition to a blockcipher encryption.
4. **Online:** a data block is processed only once.

The properties 1 and 4 contribute to small memory usage, and the properties 2 and 3 contribute to small program/circuit footprint. On top of the above properties, the fifth property regarding associated data (AD) is also important for performance:

5. **Efficient Handling of Static AD:** static AD can be precomputed.

We design a lightweight blockcipher-based AEAD mode of operation called SAEB: the first mode of operation that satisfies all the five properties to the best of our knowledge. Performance of SAEB is evaluated in various software and hardware platforms. The evaluation results show that SAEB outperforms conventional blockcipher-based AEAD modes of operation in various performance metrics for lightweight cryptography.

Keywords: Lightweight cryptography · authenticated encryption with associated data · blockciphe · mode of operation.

1 Introduction

There is a huge demand for secure connectivity in constrained embedded devices used for pervasive computing. The technology trend has made lightweight cryptography a very active research topic in cryptography. Indeed, a number of lightweight blockciphers have been proposed so far [BBI+15, BPP+17, BSS+13, BJK+16, BCG+12, GPPR11, SIH+11, SMMK13] and PRESENT [BKL+07] and CLEFIA [SSA+07] are standardized in ISO/IEC 29192-2.

A blockcipher is usually used within a *mode of operation.* Notably, authenticated encryption with associated data (AEAD) that provides both confidentiality and authenticity is widely used. Blockcipher-based AEAD schemes (or modes of operation) have been studied over the last two decades. As a result, blockcipher-based AEAD schemes such as AES-GCM [Dwoa] and AES-CTR [Dwob] with HMAC-SHA256 [Dan] are used as de facto standard. However, the research is being pushed forward even today along with the CAESAR competition [CAE] aiming to make a portfolio of AEAD schemes that have advantages over the conventional ones.

Based on the above background, study on lightweight AEAD mode of operation is emerging. However, the study of lightweight AEAD mode of operation is less mature compared to that of lightweight blockcipher. Although some lightweight AEAD schemes have been submitted to the CAESAR competition, most of them use dedicated construction. In practice, there is a number of cases wherein a blockcipher-based AEAD mode of operation is needed. For example, more and more computing platforms are having accelerators for existing blockciphers such as AES. Moreover, most of the countermeasures against side-channel analysis and fault analysis are studied considering such blockciphers. Efficient implementation of such countermeasures for a dedicated AEAD scheme is not trivial [MPL+11]. Therefore, lightweight blockcipher-based AEAD mode of operation is studied and a new mode of operation called `SAEB` is proposed in the paper.

1.1 Design Goal

Although the term "lightweight" is not strictly defined, it usually refers to a primitive that allows a compact implementation in a target platform. That can be a small program/RAM footprint in software implementation and compact circuit/register area in hardware implementation. Meanwhile there are conventionally known good properties of AEAD schemes (e.g., online and inverse-free). All the good properties cannot be satisfied at the same time because some properties are incompatible. Therefore, choosing good properties considering trade-offs is essential in designing an AEAD mode of operation. What is the properties desirable for lightweight AEAD mode of operation? In the current study, we prioritized the following four properties in designing `SAEB`.

1. **Minimum State Size:** State size is the size of memory needed for storing internal values for an AEAD scheme. That is important because the state size directly corresponds to the size of memory (i.e., RAM and register in software and hardware, respectively). When a block size of an underlying blockcipher is n bits, the state sizes of AEAD modes of operation should be greater than or equal to n bits. Hence, the minimum state size is n bits.

2. **Inverse Free:** A blockcipher-based scheme is said to be inverse free if it does not call a blockcipher decryption. The property contributes to reduce resource consumption (e.g., ROM and/or circuit) because the blockcipher decryption can be omitted.

3. **XOR Only:** Blockcipher-based AEAD scheme is composed of a blockcipher and extra operations. For example, `AES-GCM` requires XOR and `GHASH` in addition to AES encryption. A scheme is said to be *XOR Only* if the extra operation is XOR only. We consider this conceptually minimum because XOR is a basic operation even needed for basic modes of operation such as `CTR` and `CBC`. The property is also practical. A common strategy for compact implementation is to share ROM/RAM/gate between operations. XOR is suitable for resource sharing because it is frequently used within blockciphers and is bitwise operation.

4. **Online:** A scheme is said to be online if a data block is processed only once. Otherwise a data buffer (i.e., memory) is needed to store intermediate results for later use. Memory resource needed for the data buffer can dominate the size of an implementation because the data buffer should be as large as a maximum possible data length. In case of Ethernet, the data size can be as large as 1500 bytes.

On top of the above properties, the fifth property regarding associated data (AD) is added:

5. **Efficient Handling of Static AD:** In some use cases, the same AD is used for every encryption procedure, i.e., AD is static. A scheme is efficient regarding static AD if an intermediate result on static AD can be reused. In that case, static AD is processed only once in the precomputation phase thereby reducing computational load [Rog02]. Indeed, `AES-GCM` and many other blockcipher-based AEAD schemes satisfy this property.

In summary, our goal is to design an AEAD mode of operation that satisfies all the above properties.

1.2 Our Contribution

In this paper, we present the mode of operation `SAEB` (which stands for **S**mall (**S**imple, **S**lim, **S**ponge-based) **AE**AD from **B**lockcipher). That is the first mode of operation that satisfies the five good properties at the same time.

1.2.1 Design

`SAEB` is a nonce-based AEAD mode of operation, and its design follows the sponge-based design methodology [BDPA08, BDPA12a, BDPA12b]; the technique conventionally used with permutation is applied to blockcipher. `SAEB` can be thought as a cascaded n-bit blockcipher. Data block (either for AD, nonce, plaintext, or ciphertext) is absorbed in between encryptions with XOR. Hence, the properties 3 and 4 are satisfied. AD, nonce, and plaintext are fed in the order. Since AD is fed before nonce, static AD can be processed in advance, and thus the property 5 is satisfied. Moreover, the state size of `SAEB` is n bits and a blockcipher decryption is not used, thus the properties 1 and 2 are satisfied.

1.2.2 Security and Parameter Choice

We prove that `SAEB` achieves the birthday-bound security. The dominant terms in our bound are $(\sigma_A + \sigma_{\mathcal{D}})/2^c$ and $(\sigma_{\mathcal{E}} + \sigma_{\mathcal{D}})^2/2^n$, where r is the length of a plaintext block, $c = n - r$, $\sigma_{\mathcal{E}}$ is the number of all data blocks (AD blocks and plaintext blocks) by encryption queries, $\sigma_{\mathcal{D}}$ is the number of all data blocks (AD blocks and ciphertext blocks) by decryption queries, and σ_A is the number of all AD blocks by encryption queries. By setting $c = r = n/2$, `SAEB` achieves the birthday-bound security. On the other hand, since c and $\sigma_{\mathcal{E}}$ are independent, if $\sigma_A + \sigma_{\mathcal{D}} \ll 2^{n/2}$ (e.g., $\sigma_A \ll 2^{n/2}$ is satisfied for static AD; $\sigma_{\mathcal{D}} \ll 2^{n/2}$ is satisfied when the number of forgeries by decryption queries is limited by a system), the parameter c can be shorter than $n/2$ bits, that is, the efficiency of the procedure to take plaintext blocks can be improved, while keeping the birthday-bound security.

1.2.3 Comparison

Table 1 summarizes the five properties discussed in Subsection 1.1 satisfied in `SAEB` and existing blockcipher-based AEAD modes of operation, namely `GCM` [MV04], `OCB` [RBBK01, Rog04, KR11], `OTR` [Min14], `CLOC` [IMGM14], `JAMBU` [WH], and `COFB` [CIMN17]. The table shows that only `SAEB` satisfies all the properties. Especially, only `SAEB` achieves the property 1. That outperforms conventionally smallest state size of $3n/2$ bits achieved by `JAMBU` [WH] and `COFB` [CIMN17].

It would be fair to mention properties not satisfied in `SAEB`: efficiency and parallelizability. They are undoubtedly important properties but not chosen because the five properties are prioritized considering the purpose of a lightweight AEAD mode of operation. `OCB`, `OTR` and `COFB` are rate-1, that is, a blockcipher is called only once for each n-bit data block. Others involving `SAEB` require more than one blockcipher calls. Therefore,

Table 1: Comparison between SAEB and existing AEAD modes of operation GCM [MV04], OCB [KR11, Rog04, RBBK01], OTR [Min14], CLOC [IMGM14], JAMBU [WH] and COFB [CIMN17] with respect to the properties 1-5. "✓" (resp., "—") means that the corresponding property is satisfied (resp., not satisfied). The line with "Min. State Size" considers the property 1, where the inernal state sizes are given. The line with "Inv. Free" considers the property 2. The line with "XOR Only" considers the property 3. The line with "Online" considers the property 4. The line with "Static AD" considers the property 5.

Scheme	Min. State Size	Inv. Free	XOR Only	Online	Static AD
GCM	— ($4n$ bits)	✓	—	✓	✓
OCB	— ($3n$ bits)	—	—	✓	✓
OTR	— ($4n$ bits)	✓	—	✓	✓
CLOC	— ($2n$ bits)	✓	✓	✓	✓
JAMBU	— ($3n/2$ bits)	✓	✓	✓	—
COFB	— ($3n/2$ bits)	✓	—	✓	—
SAEB	✓ (n bits)	✓	✓	✓	✓

rate-1 schemes are more efficient in terms of asymptotic throughput. It is worth noting that CLOC and SAEB can be more efficient for short-length data likely to be used in constrained devices. That is because the rate-1 modes of operation require high initial cost due to precomputation. The experimental result shown in [IMGM14] indicates that CLOC outperforms other schemes up to data size around 100 bytes. Since the block size of SAEB is equal to or longer than CLOC, the same result is expected for SAEB. In addition, SAEB, CLOC, JAMBU and COFB and are not parallelizable, i.e., its computational latency cannot be simply reduced by parallelizing an implementation. The property is not prioritized because that is mainly for high-performance application and not for constrained devices.

1.2.4 Software Implementation on a Low-end Microcontroller

SAEB can be implemented with an extremely small footprint in software on embedded environments. We demonstrate this on Renesas's RL78 low-end microcontroller [Ren]. In fact, by adding 20 RAM bytes and 200 ROM bytes to any 128-bit block cipher, we can design full SAEB software with the underlying blockcipher, including the procedures to handle AD blocks, plaintext/ciphertext blocks and tag generation/verification. We believe that this is the smallest AEAD mode of operation ever published.

1.2.5 Hardware Implementation

SAEB also achieves small circuit area in hardware implementation. The performance is evaluated in various platforms namely ASIC (*NanGate* 45-nm CMOS [Nan]) and FPGA (*Xilinx Virtex-7*, and *Intel Cyclone V*). In ASIC, SAEB can be implemented with an extra 832 [GE] in addition to a blockcipher implemenation. If it is combined with a conventional compact AES implementation [MPL$^+$11], the total circuit area is $3,502$ [GE] which outperforms conventional implementations of CLOC, SILC, and AES-OTR given in [BBM16]. The same circuit for SAEB with AES can be realized using 348 [LUTs] with 242 [FFs] on *Virtex-7* and 299 [ALMs] with 127 [FFs] on *Cyclone V*. That outperforms conventional FPGA implementations of COFB, ACORN, JAMBU, and ASCON evaluated in the same FPGA chips [CIMN17][GMU].

1.3 Further Related Work

To our knowledge, existing sponge-based AEAD schemes based on permutations satisfying the property 5 were given in only [BDPA12a].[1] The security bound, which is based on the indifferentiability of the sponge function [BDPA08], is roughly $\sigma^2/2^s$, where σ is the number of data blocks by all queries and s is the permutation size minus the size of a data block for each permutation call. Hence, applying the security bound to `SAEB` (blockcipher setting), the resultant bound does not achieve the birthday one with respect to the block size n. Other sponge-based AEAD schemes based on permutations [BDPA12a, BDPA12b, JLM14, DMA17] do not satisfy the property 5, because nonce is processed before AD blocks.

On the other hand, using the Even-Mansour scheme [DKS12, EM91, EM97], `SAEB` can be converted from the blockcipher setting into the permutation one.[2] This conversion offers the birthday term with respect to the permutation size from the security bound of the Even-Mansour scheme, and the resultant bound is the same as the best bound of the permutation-based sponge-based AEAD schemes [JLM14], while the resultant AEAD scheme satisfies the property 5.

Finally, note that unlike a blockcipher, a permutation has no key scheduling, that is, the memory to keep the subkey is not required. Hence, using a permutation with size $n + k$ or less (k is the subkey size of the blockcipher), the memory size of a permutation-based AEAD scheme might be improved over `SAEB` without reducing the security level. However, as discussed earlier, there are still many use cases in which blockcipher-based scheme is preferred.

2 Preliminaries

2.1 Notations

Let $\{0,1\}^*$ be the set of all bit strings, $\{0,1\}^n$ the set of n-bit strings, λ an empty string, and $\emptyset$ an empty set. Let $[i] := \{1, 2, \ldots, i\}$ for a positive integer i. $\mathbf{i}_c$ denotes a c-bit representation of i where c and i are positive integers. For a finite set $\mathcal{X}$, $x \twoheadleftarrow \mathcal{X}$ means that an element is randomly drawn from $\mathcal{X}$ and is assigned to x. For a bit string x and a set $\mathcal{X}$, we denote by $|x|$ and $|\mathcal{X}|$ the bit length of x and the number of elements in $\mathcal{X}$, respectively. For a bit string x and a positive integer l such that $l < |x|$, $[x]^l$ and $[x]_l$ denote the first and last l-bit strings of x, respectively. For a bit string x and a positive integer r, $(x_1, \ldots, x_l) \xleftarrow{r} x$ means x is partitioned into r-bit blocks, where $x = x_1\|\cdots\|x_l$, if $|x|$ is not a multiple of r then $|x_l|$ is less than r, and if $x = \lambda$ then $l = 1$ and $x_1 = \lambda$. Let $\mathsf{Perm}(\mathcal{B})$ be the set of all permutations over a non-empty set $\mathcal{B}$. A random permutation in $\mathsf{Perm}(\mathcal{B})$ is defined as $P \twoheadleftarrow \mathsf{Perm}(\mathcal{B})$. Let $\mathsf{Func}(\mathcal{B})$ be the set of all functions from $\mathcal{B}$ to $\mathcal{B}$ for a non-empty set $\mathcal{B}$. A random function in $\mathsf{Func}(\mathcal{B})$ is defined as $R \twoheadleftarrow \mathsf{Func}(\mathcal{B})$. An adversary $\mathbf{A}$ with oracle access to $\mathcal{O}$ is denoted by $\mathbf{A}^{\mathcal{O}}$. An event that $\mathbf{A}^{\mathcal{O}}$ outputs a result y is denoted by $\mathbf{A}^{\mathcal{O}} \Rightarrow y$.

2.2 Definitions

2.2.1 Blockcipher

Let $\mathsf{BC}(\mathcal{K}, \mathcal{B})$ be the set of all encryptions of blockciphers with the set of keys $\mathcal{K}$ and the set of (plain/ciphertext) blocks $\mathcal{B}$. Fixing a blockcipher $E \in \mathsf{BC}(\mathcal{K}, \mathcal{B})$, E having a key $K \in \mathcal{K}$, denoted by $E(K, \cdot)$ or $E_K(\cdot)$, becomes a permutation over $\mathcal{B}$.

[1]In [BDPA12a], the duplex construction that becomes components of sponge-based AEAD schemes is given, and the security result offers the security bounds of the AEAD schemes.

[2]Previous works such as [CDH+12, MMH+14, ADMA15], using the Even-Mansour scheme [DKS12, EM91, EM97], proved the the security of permutation-based schemes.

We consider Pseudo-Random Permutation (PRP) security that is indistinguishability between a keyed blockcipher and a random permutation in the chosen plaintext attack setting. Let $E \in \mathsf{BC}(\mathcal{K}, \mathcal{B})$ be a blockcipher with the sets of keys $\mathcal{K}$ and blocks $\mathcal{B}$. The advantage function of a PRP-adversary $\mathbf{A}$ that outputs a bit is defined as

$$\mathbf{Adv}_E^{\mathsf{prp}}(\mathbf{A}) = \Pr[K \twoheadleftarrow \mathcal{K}; \mathbf{A}^{E_K} \Rightarrow 1] - \Pr[P \twoheadleftarrow \mathsf{Perm}(\mathcal{B}); \mathbf{A}^P \Rightarrow 1] ,$$

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

2.2.2 Nonce-Based Authenticated Encryption

A nonce-based authenticated encryption (nAE) scheme based on a blockcipher $E \in \mathsf{BC}(\mathcal{K}, \mathcal{B})$ is denoted by Π and is a pair of encryption and decryption algorithms. $\mathcal{K}, \mathcal{N}, \mathcal{M}, \mathcal{C}, \mathcal{A}$ and $\mathcal{T}$ are the sets of keys, nonces, plaintexts, ciphertexts, associated data (AD) and tags of the nAE scheme. The encryption algorithm using E_K for a key $K \in \mathcal{K}$, denoted by $\Pi.\mathtt{Enc}[E_K]$, takes a nonce $N \in \mathcal{N}$, AD $A \in \mathcal{A}$, and a plaintext $M \in \mathcal{M}$. $\Pi.\mathtt{Enc}[E_K](N, A, M)$ returns, deterministically, a pair of a ciphertext $C \in \mathcal{C}$ and a tag $T \in \mathcal{T}$. The decryption algorithm using E_K for a key $K \in \mathcal{K}$, denoted by $\Pi.\mathtt{Dec}[E_K]$, takes a tuple $(N, A, C, T) \in \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T}$. $\Pi.\mathtt{Dec}[E_K](N, A, C, T)$ returns, deterministically, either the distinguished invalid symbol $\perp$ or a plaintext $M \in \mathcal{M}$. We require $|\Pi.\mathtt{Enc}[E_K](N, A, M)| = |\Pi.\mathtt{Enc}[E_K](N, A, M')|$ when the encryption are strings and $|M| = |M'|$.

We follow the security definition in [NRS14, RS06, NRS14, GL15] that considers the indistinguishability between $(\Pi.\mathtt{Enc}[E_K], \Pi.\mathtt{Dec}[E_K])$ and $(\$, \perp)$, where $\$$ is a random-bits oracle that has the same interface as $\Pi.\mathtt{Enc}[E_K]$ and for query (N, A, M) returns a random bit string of length $|\Pi.\mathtt{Enc}[E_K](N, A, M)|$; $\perp$ is an oracle that returns the reject symbol $\perp$ for any query. Formally, we consider an adversary $\mathbf{A}$ that first interacts with either $(\Pi.\mathtt{Enc}[E_K], \Pi.\mathtt{Dec}[E_K])$ or $(\$, \perp)$, and then returns a decision bit $b \in \{0, 1\}$. The nAE-advantage function of $\mathbf{A}$ is defined as

$$\mathbf{Adv}_\Pi^{\mathsf{nAE}}(\mathbf{A}) = \Pr[K \twoheadleftarrow \mathcal{K}; \mathbf{A}^{\Pi.\mathtt{Enc}[E_K], \Pi.\mathtt{Dec}[E_K]} \Rightarrow 1] - \Pr[\mathbf{A}^{\$, \perp} \Rightarrow 1] .$$

We demand that $\mathbf{A}$ is nonce-respecting (all nonces by encryption queries are distinct), that $\mathbf{A}$ never asks a trivial decryption query (N, A, C, tag) such that there is a prior encryption query (N, A, M) with $(C, tag) = \Pi.\mathtt{Enc}[E_K](N, A, M)$, and that $\mathbf{A}$ never repeats a query.

3 Specification of SAEB

For positive integers n and k, let $E \in \mathsf{BC}(\{0,1\}^k, \{0,1\}^n)$ be the underlying blockcipher. Positive integers r_1, r_2 and r are the sizes of an AD block, a nonce and a plaintext block in SAEB respectively such that $1 \le r_1 \le n-2$, $1 \le r_2 \le n-2$, and $1 \le r \le n-2$. A positive integer τ is the tag size. Let $c_1 := n - r_1$, $c_2 := n - r_2$, and $c := n - r$. The sets $\mathcal{K}, \mathcal{N}, \mathcal{M}, \mathcal{C}, \mathcal{A}$ and $\mathcal{T}$ of SAEB are defined as $\mathcal{K} := \{0,1\}^k$, $\mathcal{N} := \{0,1\}^{r_2}$, $\mathcal{M} = \mathcal{C} := \{0,1\}^*$, $\mathcal{A} := \{0,1\}^*$, and $\mathcal{T} := \{0,1\}^\tau$. SAEB uses a one-zeros padding: for a positive integer r^* and a bit string x with $0 < |x| < r^*$, $x \| 10^*_{r^*}$ denotes a bit string where a bit 1 is appended to x, and then a minimum number of bits 0 is appended so that the length in bits becomes r^*. For example, $101 \| 10^*_8 = 10110000$ and $011 \| 10^*_4 = 0111$. SAEB is defined in Algorithm 1, and the encryption algorithm $\mathtt{SAEB}.\mathtt{Enc}[E_K]$ is illustrated in Figure 1.

Algorithm 1 `SAEB`

```
▶ Encryption $\texttt{SAEB.Enc}[E_K](N, A, M)$
1: $\mathsf{iv} \leftarrow \texttt{Hash}[E_K](N, A)$
2: $(C, T) \leftarrow \texttt{Core.Enc}[E_K](\mathsf{iv}, M)$
3: return $(C, T)$

▶ Decryption $\texttt{SAEB.Dec}[E_K](N, A, C, T)$
1: $\mathsf{iv} \leftarrow \texttt{Hash}[E_K](N, A)$
2: $(M, T') \leftarrow \texttt{Core.Dec}[E_K](\mathsf{iv}, C)$
3: if $T = T'$ then return $M$
4: if $T \neq T'$ then return $\perp$

▷ Subroutine $\texttt{Hash}[E_K](N, A)$
1: $\mathsf{sa} \leftarrow \mathbf{0}_n$; $A_1, \ldots, A_a \xleftarrow{r_1} A$
2: for $i = 1$ to $a - 1$ do $[\mathsf{sa}]^{r_1} \leftarrow [\mathsf{sa}]^{r_1} \oplus A_i$; $\mathsf{sa} \leftarrow E_K(\mathsf{sa})$
3: if $|A_a| = r_1$ then $\mathsf{sa} \leftarrow \mathsf{sa} \oplus (A_a \| \mathbf{1}_{c_1})$; $\mathsf{sa} \leftarrow E_K(\mathsf{sa})$
4: if $|A_a| < r_1$ then $\mathsf{sa} \leftarrow \mathsf{sa} \oplus ((A_a \| 10^*_{r_1}) \| \mathbf{2}_{c_1})$; $\mathsf{sa} \leftarrow E_K(\mathsf{sa})$
5: $\mathsf{iv} \leftarrow \mathsf{sa} \oplus (N \| \mathbf{3}_{c_2})$
6: return $\mathsf{iv}$

▷ Subroutine $\texttt{Core.Enc}[E_K](\mathsf{iv}, M)$
1: $\mathsf{sm} \leftarrow E_K(\mathsf{iv})$; $M_1, \ldots, M_m \xleftarrow{r} M$
2: for $i = 1$ to $m - 1$ do $[\mathsf{sm}]^r \leftarrow [\mathsf{sm}]^r \oplus M_i$; $C_i \leftarrow [\mathsf{sm}]^r$; $\mathsf{sm} \leftarrow E_K(\mathsf{sm})$
3: if $|M_m| = r$ then $\mathsf{sm} \leftarrow \mathsf{sm} \oplus (M_m \| \mathbf{1}_c)$; $C_m \leftarrow [\mathsf{sm}]^r$
4: if $|M_m| < r$ then $\mathsf{sm} \leftarrow \mathsf{sm} \oplus ((M_m \| 10^*_r) \| \mathbf{2}_c)$; $C_m \leftarrow [\mathsf{sm}]^{|M_m|}$
5: $\mathsf{sm} \leftarrow E_K(\mathsf{sm})$; $T \leftarrow [\mathsf{sm}]^\tau$
6: return $(C_1 \| C_2 \| \cdots \| C_m, T)$

▷ Subroutine $\texttt{Core.Dec}[E_K](\mathsf{iv}, C)$
1: $\mathsf{sm} \leftarrow E_K(\mathsf{iv})$; $C_1, \ldots, C_m \xleftarrow{r} C$
2: for $i = 1$ to $m - 1$ do $M_i \leftarrow [\mathsf{sm}]^r \oplus C_i$; $[\mathsf{sm}]^r \leftarrow M_i \oplus [\mathsf{sm}]^r$; $\mathsf{sm} \leftarrow E_K(\mathsf{sm})$
3: if $|C_m| = r$ then $M_m \leftarrow [\mathsf{sm}]^r \oplus C_m$; $\mathsf{sm} \leftarrow \mathsf{sm} \oplus (M_m \| \mathbf{1}_c)$
4: if $|C_m| < r$ then $M_m \leftarrow [\mathsf{sm}]^{|C_m|} \oplus C_m$; $\mathsf{sm} \leftarrow \mathsf{sm} \oplus ((M_m \| 10^*_r) \| \mathbf{2}_c)$
5: $\mathsf{sm} \leftarrow E_K(\mathsf{sm})$; $T' \leftarrow [\mathsf{sm}]^\tau$
6: return $(M_1 \| M_2 \| \cdots \| M_m, T')$
```

4 Security of SAEB

4.1 Security Bound

We prove the nAE-security of SAEB in the information-theoretic model, namely, the underlying keyed blockcipher E_K where $K \twoheadleftarrow \{0,1\}^k$ is replaced with a random permutation P where $P \twoheadleftarrow \mathsf{Perm}(\{0,1\}^n)$. The upper-bound of the nAE-security advantage is given in the following theorem. The security proof is given in Subsection 4.2. Note that using a concrete blockcipher E, the PRP-security advantage function of E is introduced in the upper-bound.

Theorem 1. *Let* $\mathbf{A}$ *be an adversary that makes* $q_\mathcal{E}$ *encryption queries and* $q_\mathcal{D}$ *decryption queries. Let* $\sigma_\mathcal{E}$ *and* $\sigma_\mathcal{D}$ *be the total numbers of blockcipher calls with distinct inputs by encryption and decryption queries, respectively. Let* σ_A *be the total number of blockcipher calls with distinct inputs in* `Hash` *in* `SAEB.Enc` *(only encryption queries). Let* σ *be the total number of blockcipher calls with distinct inputs by all queries. For any positive integer* ρ*,*

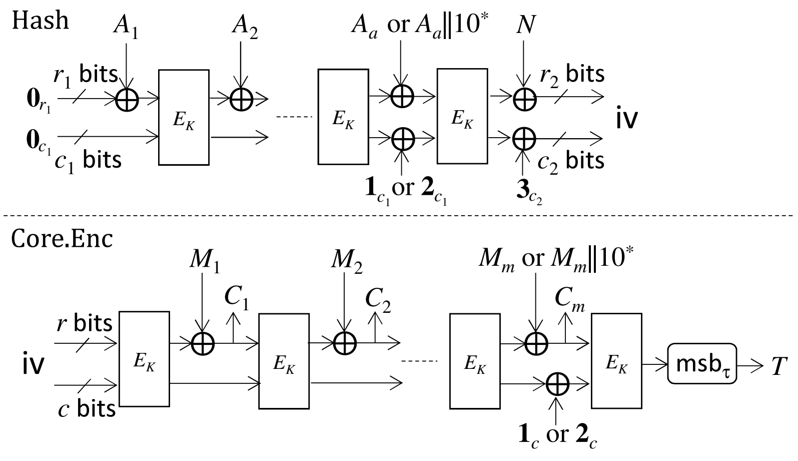

Figure 1: LightAE.Enc.

we have

$$\mathbf{Adv}^{\mathsf{nAE}}_{\mathtt{SAEB}}(\mathbf{A}) \le \frac{2\sigma^2}{2^n} + \frac{(\rho-1)(\sigma_A+\sigma_{\mathcal{D}})}{2^c} + 2^r\left(\frac{e\sigma_{\mathcal{E}}}{\rho 2^r}\right)^\rho + \frac{q_{\mathcal{D}}}{2^\tau} \enspace .$$

In the above upper-bound, the parameter c depends only on σ_A and $\sigma_{\mathcal{D}}$. The integer ρ is a free parameter and can be defined so that $\sigma_{\mathcal{E}}$ depends only on the block size n. In Section 5, we discuss how to choose these parameters.

4.2 Proof of Theorem 1

Let $\Pi :=$ SAEB. In this subsection, we upper-bound the following advantage function.

$$\mathbf{Adv}^{\mathsf{nAE}}_{\Pi}(\mathbf{A}) = \Pr[\mathsf{World}_R] - \Pr[\mathsf{World}_I] \enspace ,$$

where

$$\mathsf{World}_R := \left(P \twoheadleftarrow \mathsf{Perm}(\{0,1\}^n); \mathbf{A}^{\Pi.\mathtt{Enc}[P],\Pi.\mathtt{Dec}[P]} \Rightarrow 1\right) \text{ and } \mathsf{World}_I := \left(\mathbf{A}^{\$,\perp} \Rightarrow 1\right) \enspace .$$

In this proof, the advantage function is upper-bounded via two worlds: World_1 and World_2, which will be defined later. Then we have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{nAE}}_{\Pi}(\mathbf{A}) = &(\Pr[\mathsf{World}_R] - \Pr[\mathsf{World}_1]) + (\Pr[\mathsf{World}_1] - \Pr[\mathsf{World}_2]) + \\ &(\Pr[\mathsf{World}_2] - \Pr[\mathsf{World}_I]) \enspace . \end{aligned} \tag{1}$$

Upper-Bound of $\Pr[\mathsf{World}_R] - \Pr[\mathsf{World}_1]$

World_1 is defined as

$$\mathsf{World}_1 := \left(R \twoheadleftarrow \mathsf{Func}(\{0,1\}^n); \mathbf{A}^{\Pi.\mathtt{Enc}[R],\Pi.\mathtt{Dec}[R]} \Rightarrow 1\right) \enspace .$$

From World_R to World_1, a random permutation P is replaced with a random function R. By the PRP/PRF switching lemma [BR06], we have

$$\Pr[\mathsf{World}_R] - \Pr[\mathsf{World}_1] \le \frac{\sigma^2}{2^{n+1}} \enspace . \tag{2}$$

```
Initialization
1: $R \twoheadleftarrow \mathsf{Func}(\{0,1\}^n)$; for $\forall x \in \{0,1\}^n$ do $\mathsf{flag}[x] \leftarrow 0$
2: $\mathsf{coll}_{\mathsf{iv/sm}} \leftarrow$ false; $\mathsf{forge} \leftarrow$ false; $\mathsf{bad} := \mathsf{coll}_{\mathsf{iv/sm}} \vee \mathsf{forge}$

Encryption $\Pi^*$.Enc$(N, A, M)$
1: $\mathsf{iv} \leftarrow$ Hash$^*(N, A)$                          ▷ Given in Figure 3
2: $(T, C) \leftarrow$ Core$^*$.Enc$(\mathsf{iv}, M)$              ▷ Given in Figure 3
3: return $(T, C)$

Decryption $\Pi^*$.Dec$(N, A, C, T)$
1: $\mathsf{iv} \leftarrow$ Hash$^*(N, A)$                          ▷ Given in Figure 3
2: $(T', M) \leftarrow$ Core$^*$.Dec$(\mathsf{iv}, C)$             ▷ Given in Figure 3
3: if $\mathsf{bad} =$ false $\wedge T = T'$ then $\mathsf{forge} \leftarrow$ true
4: if $\mathsf{bad} =$ true $\wedge T = T'$ then $\boxed{\boxed{\textbf{return } M}}$
5: return $\perp$
```

Figure 2: World_1 (with double boxed statement) and World_2 (without double boxed statement).

Upper-Bound of $\Pr[\mathsf{World}_1] - \Pr[\mathsf{World}_2]$

World_2 is defined as

$$\mathsf{World}_2 := \left(\mathbf{A}^{\Pi^*.\mathtt{Enc},\Pi^*.\mathtt{Dec}} \Rightarrow 1\right),$$

where Π^*.Enc and Π^*.Dec are defined in Figure 2 and the subroutines are defined in Figure 3, where the double boxed statement is included in World_1 and removed in World_2 and the boxed statements are removed in World_1 and included in World_2. Note that Figures 2 and 3 with the double boxed statement and without the boxed statements equal World_1. In Figures 2 and 3, an array flag is introduced, where initially all entries are 0 and if $x \in \{0,1\}^n$ is input to R then $\mathsf{flag}[x]$ becomes 1. In Figure 2, events $\mathsf{coll}_{\mathsf{iv/sm}}$ and forge are introduced, and flag is used to define the event $\mathsf{coll}_{\mathsf{iv/sm}}$. $\mathsf{coll}_{\mathsf{iv/sm}} =$ **false** ensures that all inputs to R in Core*.Enc are distinct, and thus all outputs of Π^*.Enc are randomly drawn. $\mathsf{forge} =$ **false** ensures that all outputs of Π^*.Dec are $\perp$. Note that before the first query, the initialization procedure is performed. By the fundamental lemma of game-playing [BR06], we have

$$\Pr[\mathsf{World}_1] - \Pr[\mathsf{World}_2] \leq \Pr[\mathsf{bad}] \ .$$

Next, four events that will be used to upper-bound $\Pr[\mathsf{bad}]$ are defined. In these events, the following sets are used.

- L_{sa}: a multi set of all input values in sa (defined by encryption or decryption queries).
- L_{sm}: a multi set of all input values in sm.
- $L_{\mathsf{iv/sm}}$: a multi set of all input values in iv or sm defined by encryption queries (the values defined by decryption queries are not included).

Then, these events are defined below.

- $\mathsf{coll}_{\mathsf{sa}}$ is a collision event for L_{sa}:

$$\mathsf{coll}_{\mathsf{sa}} \Leftrightarrow \exists sa_1, sa_2 \in L_{\mathsf{sa}} \text{ s.t. } \overleftarrow{sa}_1 \neq \overleftarrow{sa}_2 \wedge sa_1 = sa_2,$$

where $\overleftarrow{sa}_i$ is the previous input block of sa_i for $i \in \{1,2\}$, and if sa_i is the first input block in Hash* then $\overleftarrow{sa}_i := \lambda$.

```
▷ Subroutine Hash*$(N, A)$
1: $\tilde{\mathsf{sa}} \leftarrow \mathbf{0}_n$; $A_1, \ldots, A_a \xleftarrow{r_1} A$
2: for $i = 1$ to $a - 1$ do
3:     $\mathsf{sa} \leftarrow \tilde{\mathsf{sa}} \oplus (A_i \| \mathbf{0}_{c_1})$
4:     $\tilde{\mathsf{sa}} \leftarrow R(\mathsf{sa})$
5: end for
6: if $|A_a| = r_1$ then $\mathsf{sa} \leftarrow \tilde{\mathsf{sa}} \oplus (A_a \| \mathbf{1}_{c_1})$
7: if $|A_a| < r_1$ then $\mathsf{sa} \leftarrow \tilde{\mathsf{sa}} \oplus ((A_a \| 10^*_{r_1}) \| \mathbf{2}_{c_1})$
8: $\tilde{\mathsf{sa}} \leftarrow R(\mathsf{sa})$; $\mathsf{iv} \leftarrow \tilde{\mathsf{sa}} \oplus (N \| \mathbf{3}_{c_2})$
9: return iv
```

```
▷ Subroutine Core*.Enc$(\mathsf{iv}, M)$
1: $M_1, \ldots, M_m \xleftarrow{r} M$
2: if $\mathsf{flag}[\mathsf{iv}] = 1$ then $\mathsf{coll} \leftarrow$ true
3: $\tilde{\mathsf{sm}} \leftarrow R(\mathsf{iv})$; $\mathsf{flag}[\mathsf{iv}] \leftarrow 1$; if $\mathsf{bad} =$ true then $\boxed{\tilde{\mathsf{sm}} \twoheadleftarrow \{0,1\}^n}$
4: for $i = 1$ to $m - 1$ do
5:     $\mathsf{sm} \leftarrow \tilde{\mathsf{sm}} \oplus (M_i \| \mathbf{0}_c)$; $C_i \leftarrow [\mathsf{sm}]^r$
6:     if $\mathsf{flag}[\mathsf{sm}] = 1$ then $\mathsf{coll} \leftarrow$ true
7:     $\tilde{\mathsf{sm}} \leftarrow R(\mathsf{sm})$; if $\mathsf{bad} =$ true then $\boxed{\tilde{\mathsf{sm}} \twoheadleftarrow \{0,1\}^n}$
8: end for
9: if $|M_m| = r$ then $\mathsf{sm} \leftarrow \tilde{\mathsf{sm}} \oplus (M_m \| \mathbf{1}_c)$; $C_m \leftarrow [\mathsf{sm}]^r$
10: if $|M_m| < r$ then $\mathsf{sm} \leftarrow \tilde{\mathsf{sm}} \oplus ((M_m \| 10^*_r) \| \mathbf{2}_c)$; $C_m \leftarrow [\mathsf{sm}]^{|M_m|}$
11: if $\mathsf{flag}[\mathsf{sm}] = 1$ then $\mathsf{coll} \leftarrow$ true
12: $\tilde{\mathsf{sm}} \leftarrow R(\mathsf{sm})$; if $\mathsf{bad} =$ true then $\boxed{\tilde{\mathsf{sm}} \twoheadleftarrow \{0,1\}^n}$
13: $T \leftarrow [\tilde{\mathsf{sm}}]^\tau$; return $(T, C_1 \| C_2 \| \cdots \| C_m)$
```

```
▷ Subroutine Core*.Dec$(\mathsf{iv}, C)$
1: $\tilde{\mathsf{sc}} \leftarrow R(\mathsf{iv})$; $C_1, \ldots, C_m \xleftarrow{r} C$
2: for $i = 1$ to $m - 1$ do $M_i \leftarrow [\tilde{\mathsf{sc}}]^r \oplus C_i$; $\mathsf{sc} \leftarrow \tilde{\mathsf{sc}} \oplus (M_i \| \mathbf{0}_c)$; $\tilde{\mathsf{sc}} \leftarrow R(\mathsf{sc})$
3: if $|C_m| = r$ then $M_m \leftarrow [\tilde{\mathsf{sc}}]^r \oplus C_m$; $\mathsf{sc} \leftarrow \tilde{\mathsf{sc}} \oplus (M_m \| \mathbf{1}_c)$
4: if $|C_m| < r$ then $M_m \leftarrow [\tilde{\mathsf{sc}}]^{|C_m|} \oplus C_m$; $\mathsf{sc} \leftarrow \tilde{\mathsf{sc}} \oplus (M_m \| 10^*_r \| \mathbf{2}_c)$
5: $\tilde{\mathsf{sc}} \leftarrow R(\mathsf{sc})$; $T' \leftarrow [\tilde{\mathsf{sc}}]^\tau$; return $(T', M_1 \| M_2 \| \cdots \| M_m)$
```

Figure 3: Subroutines in World_1 (without boxed statements) and World_2 (with boxed statements).

- $\mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}}$ is a collision event between L_{sa} and $L_{\mathsf{iv}/\mathsf{sm}}$:

$$\mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}} \Leftrightarrow \exists sa \in L_{\mathsf{sa}}, sm \in L_{\mathsf{iv}/\mathsf{sm}} \text{ s.t. } sa = sm.$$

- mcoll is a ρ-multi-collision event in the r-bit parts of values in L_{sm}:

$$\mathsf{mcoll} \Leftrightarrow \exists sm_1, sm_2, \ldots, sm_\rho \in L_{\mathsf{sm}} \text{ s.t. } [sm_1]^r = [sm_2]^r = \cdots = [sm_\rho]^r.$$

Hence, the following analyses consider the five events $\mathsf{coll}_{\mathsf{iv}/\mathsf{sm}}$, forge, $\mathsf{coll}_{\mathsf{sa}}$, $\mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}}$ and mcoll. Hereafter, for each event $\mathsf{x} \in \{\mathsf{coll}_{\mathsf{iv}/\mathsf{sm}}, \mathsf{forge}, \mathsf{coll}_{\mathsf{sa}}, \mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}}, \mathsf{mcoll}\}$, we upper-bound the probability that x occurs as long as other events $\{\mathsf{coll}_{\mathsf{iv}/\mathsf{sm}}, \mathsf{forge}, \mathsf{coll}_{\mathsf{sa}}, \mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}}, \mathsf{mcoll}\} \backslash \{\mathsf{x}\}$ have not occurred. The probability is denoted by $\Pr[\mathsf{x}]$. Then we have

$$\Pr[\mathsf{bad}] \leq \Pr[\mathsf{coll}_{\mathsf{iv}/\mathsf{sm}}] + \Pr[\mathsf{forge}] + \Pr[\mathsf{coll}_{\mathsf{sa}}] + \Pr[\mathsf{coll}_{\mathsf{sa},\mathsf{iv}/\mathsf{sm}}] + \Pr[\mathsf{mcoll}] \ .$$

Let $\sigma_{\mathcal{D},A}$ be the total number of blockcipher calls with distinct inputs in Hash in Π^*.Dec (only decryption queries).

• Upper-Bound of $\Pr[\mathsf{coll}_{\mathsf{sa}}]$

$\Pr[\mathsf{coll}_{\mathsf{sa}}]$ is upper-bounded: the probability that $\mathsf{coll}_{\mathsf{sa}}$ occurs as long as other events have not occurred.

Firstly, note that values in L_{sa} are defined as

$$sa[1] = A_1 \| \mathbf{0}_{c_1} \quad \text{(first block)} \tag{3}$$

$$sa[i] = (A_i \| const[i]) \oplus R(sa[i-1]) \quad \text{(i-th block with $i \neq 1$)} \tag{4}$$

where A_i is the i-th associated data block (a padding value is included at the last block), and $const[i]$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c_1}$ or $\mathbf{2}_{c_1}$ (last block). Let $sa[0] := \lambda$. Fix two elements $sa[i], sa'[j] \in L_{\mathsf{sa}}$ such that $sa[i-1] \neq sa'[j-1]$, and evaluate the probability that $sa[i] = sa'[j]$ is satisfied. Let A_i and A'_j be the AD blocks corresponding with $sa[i]$ and $sa'[j]$, respectively. Note that if $sa[i-1] = \lambda$ (i.e., $i = 1$) then $sa'[j-1] \neq \lambda$, and vice versa. Without loss of generality, assume that $sa[i-1] \neq \lambda$, that is, $i \neq 1$. By $\mathsf{forge} = \mathbf{false}$, all outputs of Π^*.`Dec` are $\perp$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \mathbf{false}$, $sa[i-1]$ is distinct from all input values in $L_{\mathsf{iv/sm}}$. Thus $R(sa[i-1])$ is drawn independently of all outputs of Π^*.`Enc` and of Π^*.`Dec`, and $\mathbf{A}$ defines A_i and A'_j without $R(sa[i-1])$. Since $R(sa[i-1])$ is randomly drawn from $\{0,1\}^n$, the probability that $sa[i] = sa'[j]$ is at most $1/2^n$.

Since the number of elements in L_{sa} is at most $\sigma_A + \sigma_{\mathcal{D},A}$, we have

$$\Pr[\mathsf{coll}_{\mathsf{sa}}] \leq \binom{\sigma_A + \sigma_{\mathcal{D},A}}{2} \cdot \frac{1}{2^n} \leq \frac{(\sigma_A + \sigma_{\mathcal{D},A})^2}{2^{n+1}} \; .$$

• Upper-Bound of $\Pr[\mathsf{coll}_{\mathsf{sa,iv/sm}}]$

$\Pr[\mathsf{coll}_{\mathsf{sa,iv/sm}}]$ is upper-bounded: the probability that $\mathsf{coll}_{\mathsf{sa,iv/sm}}$ occurs as long as other events have not occurred.

Firstly, note that values in L_{sa} are defined as the equations (3) and (4) (see the analysis of $\mathsf{coll}_{\mathsf{sa}}$). Secondly, note that values in $L_{\mathsf{iv/sm}}$ are inputs to R in `Core`*.`Enc` and defined as

$$sm'[1] = (N' \| \mathbf{3}_{c_2}) \oplus R(sa'[a]) \quad \text{(first block),} \tag{5}$$

$$sm'[j] = (M'_{j-1} \| const'[j]) \oplus R(sm'[j-1]) \quad \text{(j-th blocks with $j \neq 1$)} \tag{6}$$

where N' is the nonce, M'_{j-1} is the $(j-1)$-th plaintext block (a padding value is included at the last block), $sa'[a]$ is the value in sa at the last block in `Hash`*, and $const'[j]$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c_1}$ or $\mathbf{2}_{c_1}$ (last block). Note that $sm'[1]$ is a value defined in iv. Hence, $\mathsf{coll}_{\mathsf{sa,iv/sm}}$ implies that one of the following cases occurs: (a) $sa[1] = sm'[1]$; (b) $sa[1] = sm'[j]$ $(j \neq 1)$; (c) $sa[i] = sm'[1]$ $(i \neq 1)$; (d) $sa[i] = sm'[j]$ $(i \neq 1, j \neq 1)$.

- (a) is considered, i.e., fixing $sa[1]$ and $sm'[1]$, the probability that $sa[1] = sm'[1]$ is upper-bounded. By $\mathsf{forge} = \mathbf{false}$, all outputs of Π^*.`Dec` are $\perp$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \mathbf{false}$, $sa'[a]$ (see (5)) is distinct from all values in $L_{\mathsf{iv/sm}}$. Thus $R(\mathsf{sa}'[a])$ is drawn independently of all outputs of Π^*.`Enc` and of Π^*.`Dec`, and $\mathbf{A}$ defines A_1 and N' without $R(sa'[a])$. Since $R(sa'[a])$ is randomly drawn from $\{0,1\}^n$, the probability that $sa[1] = sm'[1]$ is at most $1/2^n$.

- (b) is considered, i.e., fixing $sa[1]$ and $sm'[j]$, and the probability that $sa[1] = sm'[j]$ is upper-bounded. Note that this probability is upper-bounded by the one that $[sa[1]]_c = [sm'[j]]_c$. Since $[R(sm'[j-1])]_c$ is randomly drawn from $\{0,1\}^c$, the probability that $[sa[1]]_c = [sm'[j]]_c$ is at most $1/2^c$.

- (c) is considered, i.e., fixing $sa[i]$ and $sm'[1]$, the probability that $sa[i] = sm'[1]$ is upper-bounded. Note that by $const[i] \neq \mathbf{3}_{c_2}$,

$$sa[i] = sm'[1] \Leftrightarrow R(sa[i-1]) \oplus R(sa'[a]) = (A_i \| const[i]) \oplus (N' \| \mathbf{3}_{c_2}) \neq 0^n.$$

Hence, $sa[i-1] \neq sa'[a]$ is satisfied, and $R(sa[i-1])$ and $R(sa'[a])$ are independently drawn. By $\mathsf{forge} = \textbf{false}$, all outputs of $\Pi^*.\mathtt{Dec}$ are $\bot$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \textbf{false}$, $sa'[a]$ is distinct from all values in $L_{\mathsf{iv/sm}}$. Thus $R(sa'[a])$ are drawn independently of all outputs of $\Pi^*.\mathtt{Enc}$ and of $\Pi^*.\mathtt{Dec}$, and $\mathbf{A}$ defines A_i and N' without $R(sa'[a])$. Since $R(sa'[a])$ is randomly drawn from $\{0,1\}^n$, the probability that $sa[i] = sm'[1]$ is at most $1/2^n$.

- (d) is considered, i.e., fixing $sa[i]$ and $sm'[j]$, the probability that $sa[i] = sm'[j]$ is upper-bounded. The collision probability is upper-bounded by the one that $[\mathsf{sa}[i]]_c = [\mathsf{sm}'[j]]_c$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \textbf{false}$, $sa[i-1] \neq sm'[j-1]$, thus $R(sa[i-1])$ and $R(sm'[j-1])$ are independently drawn, and the probability that $[sa[i]]_c = [sm'[j]]_c$ is at most $1/2^c$.

Regarding (b) and (d), by $\mathsf{mcoll} = \textbf{false}$, for $sa[i] \in L_{\mathsf{sa}}$, the number of values in L_{sm} whose first r-bit values equal $[sa[i]]^r$ is at most $\rho - 1$. Hence, fixing $sa[i] \in L_{\mathsf{sa}}$, the probability that for some $sm \in L_{\mathsf{iv/sm}}$ $sa[i] = sm$ is at most $(\rho-1)/2^c$.

Finally, since there are at most $\sigma_A + \sigma_{\mathcal{D},A}$ distinct values in L_{sa}, we have

$$\Pr[\mathsf{coll}_{\mathsf{sa,iv/sm}}] \leq \frac{(\rho-1)(\sigma_A + \sigma_{\mathcal{D},A})}{2^c} + \frac{q_{\mathcal{E}}(\sigma_A + \sigma_{\mathcal{D},A})}{2^n} \ .$$

• Upper-Bound of $\Pr[\mathsf{coll}_{\mathsf{iv/sm}}]$

$\Pr[\mathsf{coll}_{\mathsf{iv/sm}}]$ is upper-bounded: the probability that $\mathsf{coll}_{\mathsf{iv/sm}}$ occurs as long as other events have not occurred.

Firstly, note that values defined at the α-th encryption query in $L_{\mathsf{iv/sm}}$ are defined as

$$\begin{aligned} sm^\alpha[1] &= (N^\alpha \| \mathbf{3}_{c_2}) \oplus R(sa^\alpha[a_\alpha]) && \text{(first block)},\\ sm^\alpha[i] &= (M^\alpha_{i-1} \| const^\alpha[i]) \oplus R(sm^\alpha[i-1]) && \text{(i-th blocks with $i \neq 1$)} \end{aligned}$$

where $\alpha \in [q_{\mathcal{E}}]$, N^α is the nonce, M^α_{i-1} is the $(i-1)$-th plaintext block (a padding value is included at the last block), $sa^\alpha[a_\alpha]$ is the value in sa at the last block in $\mathtt{Hash}^*$, and $const^\alpha[i]$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c_1}$ or $\mathbf{2}_{c_1}$ (last block). Note that $sm^\alpha[1]$ is a value defined in iv. Then, for $\alpha, \beta \in [q_{\mathcal{E}}]$, $\mathsf{coll}_{\mathsf{iv/sm}}$ consists of three collision cases: (a) $sm^\alpha[1] = sm^\beta[1]$ with $\alpha \neq \beta$; (b) $sm^\alpha[1] = sm^\beta[i]$ with $i \neq 1$; (c) $sm^\alpha[i] = sm^\beta[j]$ with $(\alpha, i) \neq (\beta, j)$, $i \neq 1$ and $j \neq 1$.

- (a) is considered, i.e., fixing $sm^\alpha[1]$ and $sm^\beta[1]$, the probability that $sm^\alpha[1] = sm^\beta[1]$ is upper-bounded. Note that by $N^\alpha \neq N^\beta$,

  $$sm^\alpha[1] = sm^\beta[1] \Leftrightarrow R(sa^\alpha[a_\alpha]) \oplus R(sa^\beta[a_\beta]) = (N^\alpha \oplus N^\beta) \| \mathbf{0}_{c_2} \neq 0^n.$$

  Thus, $\mathsf{sa}^\alpha[a_\alpha] \neq \mathsf{sa}^\beta[a_\beta]$ and $R(\mathsf{sa}^\alpha[a_\alpha])$ and $R(\mathsf{sa}^\beta[a_\beta])$ are independently drawn. By $\mathsf{forge} = \textbf{false}$, all outputs of $\Pi^*.\mathtt{Dec}$ are $\bot$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \textbf{false}$, $sa^\alpha[a_\alpha]$ is distinct from all values in iv or sm. Thus $R(sa^\alpha[a_\alpha])$ is drawn independently of all outputs of $\Pi^*.\mathtt{Enc}$ and of $\Pi^*.\mathtt{Dec}$, and $\mathbf{A}$ defines N^α and N^β without $R(sa^\alpha[a_\alpha])$. Since $R(sa^\alpha[a_\alpha])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^\alpha[1] = sm^\beta[1]$ is at most $1/2^n$.

- (b) is considered, i.e., fixing $sm^\alpha[1]$ and $sm^\beta[i]$, the probability that $sm^\alpha[1] = sm^\beta[i]$ is upper-bounded. Note that

  $$sm^\alpha[1] = sm^\beta[i] \Leftrightarrow R(sa^\alpha[a_\alpha]) \oplus R(sm^\beta[i-1]) = (N^\alpha \| \mathbf{0}_{c_2}) \oplus (M^\beta_{i-1} \| const^\beta[i]).$$

  By $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \textbf{false}$, $sa^\alpha[a_\alpha] \neq sm^\beta[i-1]$, and $R(sa^\alpha[a_\alpha])$ and $R(sm^\beta[i-1])$ are independently drawn. Similar to (a), by $\mathsf{forge} = \textbf{false}$ and $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \textbf{false}$, $\mathbf{A}$ defines N^α and M^β_{i-1} without $R(\mathsf{sa}^\alpha[a_\alpha])$. Since $R(\mathsf{sa}^\alpha[a_\alpha])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^\alpha[1] = sm^\beta[i]$ is at most $1/2^n$.

- (c) is considered, i.e., fixing $sm^\alpha[i]$ and $sm^\beta[j]$, the probability that $sm^\alpha[i] = sm^\beta[j]$ is upper-bounded. Without loss of generality, assume that $sm^\alpha[i]$ is defined after $sm^\beta[j]$ is defined, and $\mathsf{coll}_{\mathsf{iv/sm}}$ has not occurred until $sm^\alpha[i]$ is defined. Note that

$$\begin{aligned}&sm^\alpha[i] = sm^\beta[j]\\ &\Leftrightarrow R(sm^\alpha[i-1]) \oplus R(sm^\beta[j-1]) = (M^\alpha_{i-1} \| const^\alpha[i]) \oplus (M^\beta_{j-1} \| const^\beta[j]).\end{aligned}$$

  Since $\mathsf{coll}_{\mathsf{iv/sm}}$ has not occurred until $sm^\alpha[i]$ is defined, $sm^\alpha[i-1]$ is distinct from all values defined in iv or sm, including $sm^\beta[j-1]$. Thus, $R(sm^\alpha[i-1])$ is defined independently of $R(sm^\beta[j-1])$ and of all previous outputs of Π^*.`Enc`. By $\mathsf{forge} = \mathbf{false}$, all outputs of Π^*.`Dec` are $\perp$. Thus, $\mathbf{A}$ defines M^α_{i-1} and M^β_{j-1} without $R(sm^\alpha[i-1])$. Since $R(sm^\alpha[i-1])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^\alpha[i] = sm^\beta[j]$ is at most $1/2^n$.

Finally, since there are at most $(\sigma_\mathcal{E} - \sigma_A)$ values in $L_{\mathsf{iv/sm}}$, we have

$$\Pr[\mathsf{coll}_{\mathsf{iv/sm}}] \le \binom{\sigma_\mathcal{E} - \sigma_A}{2} \cdot \frac{1}{2^n} \le \frac{(\sigma_\mathcal{E} - \sigma_A)^2}{2^{n+1}} \ .$$

• Upper-Bound of Pr[mcoll]

Pr[mcoll] is upper-bounded: the probability that mcoll occurs as long as other events have not occurred.

Fix $sm \in \{0,1\}^r$ and $sm^{(1)}, sm^{(2)}, \dots, sm^{(\rho)} \in L_{\mathsf{sm}}$. Note that $\mathsf{sm}^{(i)}$ is defined as $sm^{(i)} = (M^{(i)} \| const_i) \oplus R(\overleftarrow{sm}^{(i)})$, where $\overleftarrow{sm}^{(i)}$ is the input at the previous block, $M^{(i)}$ is the plaintext block, and $const_i$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c_1}$ or $\mathbf{2}_{c_1}$ (last block). By $\mathsf{forge} = \mathbf{false}$, all outputs of Π^*.`Dec` are $\perp$. By $\mathsf{coll}_{\mathsf{iv/sm}} = \mathbf{false}$, $\overleftarrow{sm}^{(1)}, \dots, \overleftarrow{sm}^{(\rho)}$ are distinct, i.e, $R(\overleftarrow{sm}^{(1)}), \dots, R(\overleftarrow{sm}^{(\rho)})$ are independently drawn, and for each $j \in [\rho]$ $\mathbf{A}$ defines $M^{(j)}$ without $R(\overleftarrow{sm}^{(j)})$. Since $R(\overleftarrow{sm}^{(1)}), \dots, R(\overleftarrow{sm}^{(\rho)})$ are randomly drawn from $\{0,1\}^n$, the probability that $[sm^{(1)}]^r = [sm^{(2)}]^r = \cdots = [sm^{(\rho)}]^r = sm$ is at most $(1/2^r)^\rho$, and we have

$$\Pr[\mathsf{mcoll}] \le 2^r \binom{\sigma_\mathcal{E}}{\rho} \left(\frac{1}{2^r}\right)^\rho \le 2^r \left(\frac{e\sigma_\mathcal{E}}{\rho 2^r}\right)^\rho ,$$

where Stirling's approximation is used ($\rho! \ge (\rho/e)^\rho$ for any ρ).

• Upper-Bound of Pr[forge]

Pr[forge] is upper-bounded: the probability that forge occurs as long as other events have not occurred. In this evaluation, the following two cases are considered. Let L_{sc} be the set of all inputs values in sc at the last block in `Core`*.`Dec` (defined at the step 3 or 4 in `Core`*.`Dec`).

(A) $\forall sc \in L_{\mathsf{sc}}, sm \in L_{\mathsf{iv/sm}}$ such that sm is defined before sc is defined: $sc \ne sm$.

(B) $\exists sc \in L_{\mathsf{sc}}, sm \in L_{\mathsf{iv/sm}}$ such that sm is defined before sc is defined and $sc = sm$.

Hence, forge can be divided into the following two cases.

- The case (A) has been satisfied (i.e., the case (B) has not occurred) and then forge becomes **true**.
- The case (B) occurs and then forge becomes **true**.

Hence, Pr[forge] is upper-bounded by the sum of the probabilities that the first case occurs and that the second case occurs. Note that the probability for the first case is upper-bounded by Pr[forge|(A)] and the probability for the second case is upper-bounded by the probability that the case (B) occurs as long as forge = **false**, i.e., Pr[(B)|¬forge]. Thus, we have

$$\Pr[\mathsf{forge}] \leq \Pr[\mathsf{forge}|(A)] + \Pr[(B)|\neg\mathsf{forge}] \ .$$

For Pr[forge|(A)], the case (A) ensures that all tags T' defined in Π^*.Dec are drawn independently of all outputs of Π^*.Enc, and thus we have $\Pr[\mathsf{forge}|(A)] \leq q_{\mathcal{D}}/2^{\tau}$. Hereafter, $\Pr[(B)|\neg\mathsf{forge}]$ is upper-bounded.

Firstly, note that values at the α-th encryption query in $L_{\mathsf{iv/sm}}$ are defined as

$$\begin{aligned} sm^{\alpha}[1] &= (N^{\mathcal{E},\alpha}\|\mathbf{3}_{c_2}) \oplus R(sa^{\alpha}[a_{\alpha}]) && \text{(first block)},\\ sm^{\alpha}[i] &= (M^{\alpha}_{i-1}\|const^{\alpha}[i]) \oplus R(sm^{\alpha}[i-1]) && \text{(i-th blocks with $i \neq 1$)} \end{aligned}$$

where $N^{\mathcal{E},\alpha}$ is the nonce, M^{α}_{i-1} is the $(i-1)$-th plaintext block (a padding value is included at the last block), $sa^{\alpha}[a_{\alpha}]$ is the value in sa at the last block in Hash*, and $const^{\alpha}[i]$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c_1}$ or $\mathbf{2}_{c_1}$ (last block). Let $sm^{\alpha}[0] := sa^{\alpha}[a_{\alpha}]$. Note that $sm^{\alpha}[1]$ is a value defined in iv. Let m_{α} be the number of plaintext blocks at the α-th encryption query, i.e., $i \leq m_{\alpha}+1$.

Secondly, values in iv or sc in Core*.Dec at the β-th decryption query are defined as

$$\begin{aligned} sc^{\beta}[1] &= (N^{\mathcal{D},\beta}\|\mathbf{3}_{c_2}) \oplus R(sa^{\beta}[a_{\beta}]) && \text{(first block)},\\ sc^{\beta}[j] &= (M^{\beta}_{j-1}\|const^{\beta}[j]) \oplus R(sc^{\beta}[j-1]) && \text{(j-th blocks with $i \neq 1$)} \end{aligned}$$

where $N^{\mathcal{D},\beta}$ is the nonce, M^{β}_{j-1} is the $(j-1)$-th (decrypted) plaintext block (a padding value is included at the last block), $sa^{\beta}[a_{\beta}]$ is the value in sa at the last block in Hash*, and $const^{\beta}[j]$ is $\mathbf{0}_{c_1}$ (middle blocks); $\mathbf{1}_{c}$ or $\mathbf{2}_{c}$ (last block). Let $sc^{\beta}[0] := sa^{\beta}[a_{\beta}]$. Note that $sc^{\beta}[1]$ is a value defined in iv. Let m_{β} be the number of ciphertext blocks at the β-th decryption query, i.e., $j \leq m_{\beta}+1$.

Then, the case (B) implies that $\exists\alpha,\beta,i$ s.t. $sm^{\alpha}[i] = sc^{\beta}[m_{\beta}+1]$ and $sc^{\beta}[m_{\beta}+1]$ is defined after $sm^{\alpha}[i]$ is defined. Since **A** never asks a trivial decryption query, for $j = 0,1,2,\ldots$ searching two inputs $sm^{\alpha}[i-j]$ and $sc^{\beta}[m_{\beta}+1-j]$ at the same time, there exists j such that $sm^{\alpha}[i-j] = sc^{\beta}[m_{\beta}+1-j]$ and $sm^{\alpha}[i-(j+1)] \neq sc^{\beta}[m_{\beta}+1-(j+1)]$. The detail analysis to derive the fact is given in Appendix A. Hence, if the case (B) occurs, there exist $\alpha \in [q_{\mathcal{E}}]$, $\beta \in [q_{\mathcal{D}}]$, $i \in [m_{\alpha}+1]$, $j \in [m_{\beta}+1]$ such that one of the following collisions occurs, that is, $\Pr[(B)|\neg\mathsf{forge}]$ is upper-bounded by the sum of these collision probabilities.

- (a) $sm^{\alpha}[0] \neq sc^{\beta}[0] \wedge sm^{\alpha}[1] = sc^{\beta}[1]$
- (b) $sm^{\alpha}[0] \neq sc^{\beta}[j-1] \wedge sm^{\alpha}[1] = sc^{\beta}[j]$ with $j \neq 1$
- (c) $sm^{\alpha}[i-1] \neq sc^{\beta}[0] \wedge sm^{\alpha}[i] = sc^{\beta}[1]$ with $i \neq 1$
- (d) $sm^{\alpha}[i-1] \neq sc^{\beta}[j-1] \wedge sm^{\alpha}[i] = sc^{\beta}[j]$ with $i \neq 1$ and $j \neq 1$

These collision probabilities are upper-bounded below.

- (a) is considered. First, fixing $\alpha \in [q_{\mathcal{E}}]$ and $\beta \in [q_{\mathcal{D}}]$ with $sm^{\alpha}[0] \neq sc^{\beta}[0]$, the probability that $sm^{\alpha}[1] = sc^{\beta}[1]$ is upper-bounded. By forge = **false**, all outputs of Π^*.Dec are $\perp$. By $\mathsf{coll}_{\mathsf{sa,iv/sm}} =$ **false**, $sm^{\alpha}[0]$ $(= sa^{\alpha}[a_{\alpha}])$ is distinct from all elements in $L_{\mathsf{iv/sm}}$. Thus $R(sm^{\alpha}[0])$ is drawn independently of all outputs of Π^*.Enc and of Π^*.Dec, and **A** defines $N^{\mathcal{E},\alpha}$ and $N^{\mathcal{D},\beta}$ without $R(sm^{\alpha}[0])$. Since $R(sm^{\alpha}[0])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^{\alpha}[1] = sc^{\beta}[1]$ is at most $1/2^n$. Thus, the probability that $\exists\alpha,\beta$ s.t. $\mathsf{sm}^{\alpha}[0] \neq \mathsf{sc}^{\beta}[0]$ and $\mathsf{sm}^{\alpha}[1] = \mathsf{sc}^{\beta}[1]$ is at most $q_{\mathcal{E}}q_{\mathcal{D}}/2^n$.

- (b) is considered. This analysis is similar to (a). First, fixing $\alpha \in [q_\mathcal{E}]$, $\beta \in [q_\mathcal{D}]$ and $j \in [m_\beta + 1] \backslash \{1\}$ with $sm^\alpha[0] \neq sc^\beta[j-1]$, the probability that $sm^\alpha[1] = sc^\beta[j]$ is upper-bounded. By $\mathsf{forge} = \mathbf{false}$ and $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \mathbf{false}$, $R(sm^\alpha[0])$ is drawn independently of all outputs of Π^*.`Enc` and of Π^*.`Dec`, and $\mathbf{A}$ defines M^α_{i-1} and M^β_{j-1} without $R(sm^\alpha[0])$. Since $R(sm^\alpha[0])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^\alpha[1] = sc^\beta[j]$ is at most $1/2^n$. Thus, the probability that $\exists \alpha, \beta, j$ s.t. $sm^\alpha[0] \neq sc^\beta[j-1]$ and $sm^\alpha[1] = sc^\beta[j]$ is at most $q_\mathcal{E}(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A} - q_\mathcal{D})/2^n$.
- (c) is considered. This analysis is similar to (a). First, fixing $\alpha \in [q_\mathcal{E}]$, $\beta \in [q_\mathcal{D}]$ and $i \in [m_\alpha + 1] \backslash \{1\}$ with $sm^\alpha[i-1] \neq sc^\beta[0]$, the probability that $sm^\alpha[i] = sc^\beta[1]$ is upper-bounded. By $\mathsf{forge} = \mathbf{false}$ and $\mathsf{coll}_{\mathsf{sa,iv/sm}} = \mathbf{false}$, $R(sc^\beta[0])$ is drawn independently of all outputs of Π^*.`Enc` and of Π^*.`Dec`, and $\mathbf{A}$ defines M^α_{i-1} and $N^{\mathcal{D},\beta}$ without $R(sc^\beta[0])$. Since $R(sc^\beta[0])$ is randomly drawn from $\{0,1\}^n$, the probability that $sm^\alpha[i] = sc^\beta[1]$ is at most $1/2^n$. Thus, the probability that $\exists \alpha, \beta, i$ s.t. $sm^\alpha[i-1] \neq sc^\beta[0]$ and $sm^\alpha[i] = sc^\beta[1]$ is at most $(\sigma_\mathcal{E} - \sigma_A - q_\mathcal{E})q_\mathcal{D}/2^n$.
- (d) is considered. First, fixing $\alpha \in [q_\mathcal{E}]$, $\beta \in [q_\mathcal{D}]$, $i \in [m_\alpha + 1] \backslash \{1\}$ and $j \in [m_\beta + 1] \backslash \{1\}$ with $sm^\alpha[i-1] \neq sc^\beta[j-1]$, the probability that $sm^\alpha[i] = sc^\beta[j]$ is upper-bounded. Note that the probability that $sm^\alpha[i] = sc^\beta[j]$ is upper-bounded by the one that $[sm^\alpha[i]]_c = [sc^\beta[j]]_c$. By $sm^\alpha[i-1] \neq sc^\beta[j-1]$, $R(sm^\alpha[i-1])$ and $R(sc^\beta[j-1])$ are independently drawn. Thus the probability that $[sm^\alpha[i]]_c = [sc^\beta[j]]_c$ is at most $1/2^c$. Next fix only β, j. By $\mathsf{mcoll} = \mathbf{false}$, the number of elements for $sm^\alpha[i]$ whose first r-bit values equal $[sc^\beta[j]]^r$ is at most $\rho - 1$. Thus the probability that $\exists \alpha, i$ s.t. $[sm^\alpha[i]]_c = [sc^\beta[j]]_c$ is at most $(\rho-1)/2^c$. Finally, the probability that $\exists \alpha, \beta, i, j$ s.t. $sm^\alpha[i-1] \neq sc^\beta[j-1]$ and $[sm^\alpha[i]]_c = [sc^\beta[j]]_c$ is at most $(\rho-1)(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A} - q_\mathcal{D})/2^c$.

Summing the above upper-bounds gives

$$\Pr[(\mathrm{B})|\neg\mathsf{forge}] \leq \frac{q_\mathcal{E}(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A}) + (\sigma_\mathcal{E} - \sigma_A)q_\mathcal{D}}{2^n} + \frac{(\rho-1)(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A})}{2^c} .$$

Finally, we have

$$\begin{aligned}\Pr[\mathsf{forge}] &\leq \Pr[\mathsf{forge}|(\mathrm{A})] + \Pr[(\mathrm{B})|\neg\mathsf{forge}] \\ &\leq \frac{q_\mathcal{D}}{2^\tau} + \frac{q_\mathcal{E}(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A}) + (\sigma_\mathcal{E} - \sigma_A)q_\mathcal{D}}{2^n} + \frac{(\rho-1)(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A})}{2^c} .\end{aligned}$$

• Upper-Bound of $\Pr[\mathsf{World}_1] - \Pr[\mathsf{World}_2]$

Finally, the above upper-bounds give

$$\begin{aligned}&\Pr[\mathsf{World}_1] - \Pr[\mathsf{World}_2] \\ &\leq \frac{(\sigma_A + \sigma_{\mathcal{D},A})^2}{2^{n+1}} + \frac{(\rho-1)(\sigma_A + \sigma_{\mathcal{D},A})}{2^c} + \frac{q_\mathcal{E}(\sigma_A + \sigma_{\mathcal{D},A})}{2^n} + \frac{(\sigma_\mathcal{E} - \sigma_A)^2}{2^{n+1}} \\ &\quad + 2^r\left(\frac{e\sigma_\mathcal{E}}{\rho 2^r}\right)^\rho + \frac{q_\mathcal{E}(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A}) + (\sigma_\mathcal{E} - \sigma_A)q_\mathcal{D}}{2^n} + \frac{(\rho-1)(\sigma_\mathcal{D} - \sigma_{\mathcal{D},A})}{2^c} + \frac{q_\mathcal{D}}{2^\tau} \\ &\leq \frac{(\rho-1)(\sigma_A + \sigma_\mathcal{D})}{2^c} + \frac{(\sigma_A + \sigma_{\mathcal{D},A})^2 + (\sigma_\mathcal{E} - \sigma_A)^2}{2^{n+1}} \\ &\quad + \frac{q_\mathcal{E}(\sigma_A + \sigma_\mathcal{D}) + (\sigma_\mathcal{E} - \sigma_A)q_\mathcal{D}}{2^n} + 2^r\left(\frac{e\sigma_\mathcal{E}}{\rho 2^r}\right)^\rho + \frac{q_\mathcal{D}}{2^\tau} \\ &\leq \frac{(\rho-1)(\sigma_A + \sigma_\mathcal{D})}{2^c} + \frac{(\sigma_\mathcal{E} + \sigma_{\mathcal{D},A})^2}{2^{n+1}} + \frac{\sigma^2}{2^n} + 2^r\left(\frac{e\sigma_\mathcal{E}}{\rho 2^r}\right)^\rho + \frac{q_\mathcal{D}}{2^\tau} \\ &\leq \frac{(\rho-1)(\sigma_A + \sigma_\mathcal{D})}{2^c} + \frac{1.5\sigma^2}{2^n} + 2^r\left(\frac{e\sigma_\mathcal{E}}{\rho 2^r}\right)^\rho + \frac{q_\mathcal{D}}{2^\tau} . \qquad (7)\end{aligned}$$

Upper-Bound of $\Pr[\mathsf{World}_2] - \Pr[\mathsf{World}_I]$

In World_2, for any encryption query, the response is randomly drawn, and for any decryption query, $\perp$ is returned. Hence, World_2 is indistinguishable from World_I, that is,

$$\Pr[\mathsf{World}_2] - \Pr[\mathsf{World}_I] = 0 \ . \tag{8}$$

Conclusion of the Proof

Putting the upper-bounds (2), (7), and (8) into (1) gives

$$\mathbf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathbf{A}) \leq \frac{2\sigma^2}{2^n} + \frac{(\rho-1)(\sigma_A + \sigma_{\mathcal{D}})}{2^c} + 2^r\left(\frac{e\sigma_{\mathcal{E}}}{\rho 2^r}\right)^{\rho} + \frac{q_{\mathcal{D}}}{2^\tau} \ .$$

5 Parameters

In this section, we study how to choose parameters of SAEB.

5.1 Recommended Parameters

First we give the recommended parameters of SAEB: the lengths of an AD block, a nonce, and a plaintext block are defined as $(r_1, r_2, r) = (n-8, n/2, n/2)$. In this setting, for each n-bit plaintext block, a blockcipher is performed twice. For each n-bit AD block, a blockcipher is called $n/(n-8)$ times.

Next, we study the security bound of SAEB in Theorem 1 for the parameters. In this setting, the upper-bound of SAEB becomes

$$\frac{2\sigma^2}{2^n} + \frac{(\rho-1)(\sigma_A + \sigma_{\mathcal{D}})}{2^{n/2}} + 2^{n/2}\left(\frac{e\sigma_{\mathcal{E}}}{\rho 2^{n/2}}\right)^{\rho} + \frac{q_{\mathcal{D}}}{2^\tau} \ .$$

Putting $\rho = n/2$, the upper-bound becomes

$$\frac{2\sigma^2}{2^n} + \frac{(n-2)(\sigma_A + \sigma_{\mathcal{D}})}{2^{(n/2)+1}} + \left(\frac{2e\sigma_{\mathcal{E}}}{n 2^{n/2}}\right)^{n/2} + \frac{q_{\mathcal{D}}}{2^\tau} \ .$$

Hence, SAEB is a secure nAE scheme as long as $(\sigma, \sigma_A, \sigma_{\mathcal{D}}, q_{\mathcal{D}})$ are less than roughly $(2^{n/2}, 2^{n/2}/n, 2^{n/2}/n, 2^\tau)$, and if $\sigma_A \ll 2^{n/2}/n$, $\sigma_{\mathcal{D}} \ll 2^{n/2}/n$, $q_{\mathcal{D}} \ll 2^\tau$, then SAEB achieves birthday-bound security (security up to $2^{n/2}$ blockcipher calls).

5.2 Parameters with Better Efficiency

In the security bound of SAEB, the parameter c depends only on $\sigma_{\mathcal{D}}$ and σ_A.[3] Hence, if $\sigma_{\mathcal{D}}$ and σ_A are limited,[4] c can be reduced less than $n/2$, thereby the efficiency of Core.Enc and Core.Dec can be improved. We give parameters that cover important applications such as HTTPs, MQTT, firmware updating, etc, when using a 128-bit blockcipher, i.e., $n = 128$: the lengths of an AD block, a nonce and a plaintext block are defined so that

[3] $\sigma_{\mathcal{D}}$ is the number of blockcipher calls with distinct inputs by decryption queries made by an adversary. Hence, it does not include blockcipher calls with repeated inputs and blockcipher calls whose queries are sent from honest parties. σ_A is the number of blockcipher calls with distinct AD blocks by encryption queries made by an adversary. Hence, it does not include blockcipher calls with repeated AD blocks.

[4] Usually, AD is short and is not frequently changed i.e., σ_A is small. For example, as mentioned in [Rog02], AD is used as a packet header that is not frequently changed. Regarding the parameter $\sigma_{\mathcal{D}}$, when the number of forgery attempts $q_{\mathcal{D}}$ is limited, i.e., a key is changed when $q_{\mathcal{D}}$ reaches some threshold, $\sigma_{\mathcal{D}}$ can be small.

Table 2: Our Implementation of blockciphers on RL78.

	Version	ROM bytes	RAM bytes	Cycles/block	Unrolled rounds
`AES`	`Tiny`	399	42	8704	1/16
	`Fast`	926	20	3554	1
`SIMON`	`Tiny`	111	20	21050	1/8
	`Fast`	629	18	10836	2
`SPECK`	`Tiny`	71	39	11432	1/8
	`Fast`	309	18	4793	2

$(r_1, r_2) = (120, 64)$ and $r \leq 80$. Hereafter, the faster parameters $(r_1, r_2, r) = (120, 64, 80)$ are considered. Regarding the security bound, putting $\rho = 4$, it becomes

$$\frac{2\sigma^2}{2^{128}} + \frac{3(\sigma_A + \sigma_{\mathcal{D}})}{2^{48}} + \left(\frac{e\sigma_{\mathcal{E}}}{4 \cdot 2^{63}}\right)^4 + \frac{q_{\mathcal{D}}}{2^\tau} \ .$$

Hence, `SAEB` is a secure nAE scheme as long as $(\sigma, \sigma_A, \sigma_{\mathcal{D}}, q_{\mathcal{D}})$ are less than roughly $(2^{63}, 2^{48}/3, 2^{48}/3, 2^\tau)$, and if σ_A, $\sigma_{\mathcal{D}}$ and $q_{\mathcal{D}}$ are limited as $\sigma_A \ll 2^{48}/3$, $\sigma_{\mathcal{D}} \ll 2^{48}/3$ and $q_{\mathcal{D}} \ll 2^\tau$, `SAEB` achieves birthday-bound security. Regarding the efficiency, for each 128-bit plaintext block, a blockcipher is called 1.6 times on average, that is, regarding the procedure to take plaintext blocks, `SAEB` is performed roughly 5/4 times faster than the recommended parameters.

Next, for these parameters, we study the relation between $c = 48$ and the parameters $(\sigma_A, \sigma_{\mathcal{D}})$. For simplicity, we assume that the term $q_{\mathcal{D}}/2^\tau$ is negligible compared with other terms. Hence, we focus on the term $3(\sigma_A + \sigma_{\mathcal{D}})/2^{48}$. Regarding the term $3\sigma_A/2^{48}$, it becomes a constant when roughly $\sigma_A = 2^{46}$ (2^{50} bytes for distinct AD blocks), that is, the number of distinct AD blocks should be limited as $\sigma_A \ll 2^{46}$. Regarding the term $3\sigma_{\mathcal{D}}/2^{48}$, it becomes a constant when roughly $\sigma_{\mathcal{D}} = 2^{46}$ (2^{50} bytes for distinct ciphertext blocks given by an adversary). For simplicity, the ciphertext length ℓ in blocks is fixed for all queries. Then, the number of forgery attempts should be limited as $q_{\mathcal{D}} \ll 2^{46}/\ell$, e.g., $q_{\mathcal{D}} \ll 2^{39}$ if $\ell = 103$ (1K byte); $q_{\mathcal{D}} \ll 2^{29}$ if $\ell = 104858$ (1M byte); $q_{\mathcal{D}} \ll 2^{19}$ if $\ell = 107374183$ (1G byte); etc. Usually, a key is changed if the security bound reaches some threshold such as $1/2^{20}$ and $1/2^{32}$, e.g., for the threshold $1/2^{20}$, a key is changed when $\sigma_{\mathcal{D}} = 2^{28}/3$; for for the threshold $1/2^{32}$, a key is changed when $\sigma_{\mathcal{D}} = 2^{14}/3$.

6 Software Implementation

In this section we demonstrate that `SAEB` can be implemented with an extremely small footprint in software on a low-end microcontroller. Our target is Renesas's RL78, which is a 16-bit CISC processor widely used in industrial embedded systems [Ren]. First we implemented three blockciphers `AES`, `SIMON` and `SPECK` [BSS+13] with 128-bit block and 128-bit key on the microcontroller and measured their size and performance as shown in Table 2. For each blockcipher we designed two versions; `Tiny`, aiming at minimizing ROM size, and `Fast`, focusing on high speed with reasonable ROM size. All codes adopt on-the-fly key scheduling to minimize the size of RAM, which is usually much more expensive than ROM.

The ROM size includes code and constant value area, and the RAM size includes stack and temporary value area excluding that for key and message. The right-most column shows the number of consecutive rounds that are unrolled in an internal loop, where $1/x$ means that a single round consists of an x-time loop. All programs have been implemented within 1KB ROM and 50-byte RAM, and in particular our tiny versions attain extremely small memory size; `SIMON` and `SPECK` requires only 111 and 71 ROM bytes, respectively.

Next we designed full `SAEB` software containing `Hash`, `Core.Enc` and `Core.Dec` for all the six blockcipher versions. As a result, our implementation requires only 200 ROM bytes

Table 3: Our Implementation of SAEB with the default parameters on RL78.

Underlying	ROM	RAM	Operation	Cycles/byte for x-byte data					
blockcipher	bytes	bytes		16	32	64	128	256	∞
AES_tiny	599	68	Hash	1118	838	687	617	616	583
			Enc/Dec	1656	1380	1232	1162	1127	1093
AES_fast	1126	46	Hash	475	356	285	255	254	240
			Enc/Dec	691	576	508	479	464	449
SIMON_tiny	311	46	Hash	2662	1996	1652	1486	1484	1406
			Enc/Dec	3971	3310	2968	2802	2719	2636
SIMON_fast	829	44	Hash	1385	1038	854	767	766	725
			Enc/Dec	2056	1714	1532	1446	1403	1360
SPECK_tiny	271	65	Hash	1460	1094	901	809	808	765
			Enc/Dec	2168	1807	1616	1525	1479	1434
SPECK_fast	509	44	Hash	630	472	382	343	341	322
			Enc/Dec	923	770	682	643	624	604

for the mode of operation, independent of parameters, which means SAEB can be fully implemented with 200-byte overhead in addition to the size of its underlying blockcipher. As far as we know, this is the smallest AEAD mode in a low-end software environment.

Table 3 shows our implementation results for SAEB with the recommended parameters: $r_1 = n - 8$ and $r = n/2$. The speed was measured for the Hash part and the Core.Enc/Core.Dec part separately, where the former processes AD and the latter handles plaintext/ciphertext and tag. Our codes run exactly in the same number of cycles for encryption and decryption for any input data, and hence have resistance against timing attacks.

It is seen that SAEB with SPECK_tiny has an extremely small footprint, within 300 ROM bytes. Even SAEB with AES_tiny can be implemented within 600 ROM bytes. RAM requirement is also very small within 70 bytes for all algorithms, which is due to the design of our mode of operation and our on-the-fly implementation of underlying blockciphers. SAEB with AES_fast is the fastest with 691 cycles/byte for encryption and decryption. Meanwhile SAEB with SPECK_fast is 35% slower but 55% smaller than SAEB with AES_fast.

In Table 4, our implementations of SAEB with recommended parameters: $r_1 = n - 8$ and $r = n/2$ are compared with implementations of CLOC [IMGM14] and OCB [KR11] evaluated in the previous work [IMGM14]. The previous work uses another low-end 8-bit microcontroller ATmega128 as an evaluation platform. Note that in general a program on RISC ATmega128 processor tends to be larger and faster than that on CISC RL78 processor. Since [IMGM14] measured the speed with 16-byte fixed length AD, SAEB is implemented under the same condition. Also, the numbers of cycles for initialization and message processing measured separately in [IMGM14] are summed in Table 4 for comparison.

Table 4 clearly shows that SAEB outperforms CLOC and OCB in ROM and RAM sizes. Processing speed of SAEB outperforms that of the rate-1 AEAD scheme OCB up to 32-byte data. Compared to CLOC, SAEB requires more cycles to process the same amount of data although the number of blockcipher calls for message is the same. The difference is caused by the different number of blockcipher calls for AD: SAEB needs two blockcipher calls to process 16-byte AD because of its block size for AD (15 bytes). That is a boundary condition, and the difference is expected to be smaller with other conditions e.g., 15-byte or 17-byte AD or to be removed for static AD. Note that AES_fast is implemented with on-the-fly key scheduling. That means our implementation has a room for significant speed improvement by separating the key scheduling part in return for additional RAM requirement, specifically 160/528/240 additional bytes for AES/SIMON/SPECK, respectively, and small additional ROM bytes.

Finally, SAEB is compared with the state-of-the-art lightweight blockcipher-based AEAD

Table 4: Performance comparison with existing AEAD schemes.

	Mode of operation	Underlying blockcipher	ROM bytes	RAM bytes	Cycles/byte for x-byte data					
					16	32	64	128	256	∞
[IMGM14]	`CLOC`	`AES`	2980	362	875	612	480	414	381	-
[IMGM14]	`OCB-E`	`AES`	5010	971	1527	891	573	414	334	-
[IMGM14]	`OCB-D`	`AES`	5010	971	1562	928	611	453	374	-
This work	`SAEB`	`AES_fast`	1126	46	1166	813	626	538	493	449

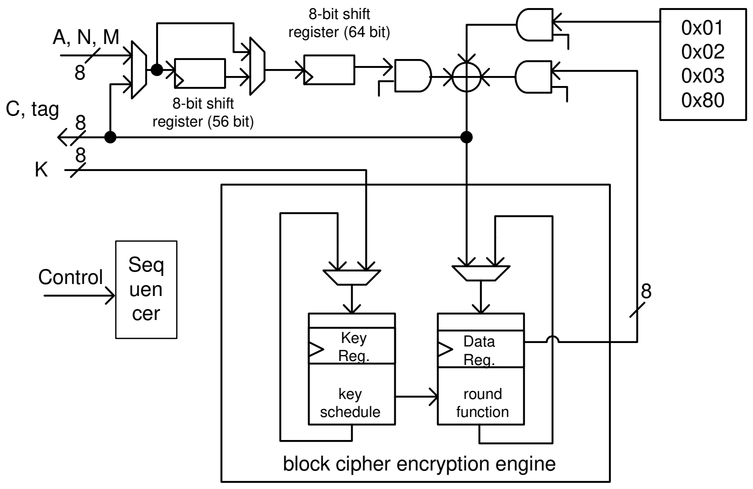

Figure 4: A lightweight hardware architecture for `SAEB`.

`COFB` [CIMN17]. Although no software implementation is given for `COFB` [CIMN17], `SAEB` is expected to achieve smaller RAM size thanks to the small state size as shown in Table 1.

7 Hardware Implementation

In this section, lightweight hardware architecture of `SAEB` is shown. We consider a concrete implementation using `AES` as a blockcipher, however, its extensions to other blockciphers (e.g., `SIMON` and `SPECK`) should be easy.

Figure 4 shows the proposed architecture. There is a 8-bit bus for feeding AD A, nonce N, and message M. A, N, and M are distinguished by a 2-bit control signal. In addition, there is another 1-bit control signal indicating the final block of A and M. The final blocks are padded automatically in hardware side. In addition, there is another 8-bit bus dedicated for key K.

In Figure 4, a circuit block indicated by "blockcipher encryption engine" is a lightweight `AES` implementation based on [MPL$^+$11] except that decryption capability is removed. The `AES` circuit has 8-bit input and output buses. By feeding input bytes and getting output bytes synchronously, the maximum throughput of "block cipher encryption engine" can be achieved. There are 56-bit and 64-bit shift registers used to temporarily store incoming byte stream for A, N, and M, and feed them to the `AES` circuit synchronously. Note that the shift registers are not strictly a part of `SAEB`, and thus can be removed from the design in order to minimize the circuit area within the IP. However, having the shift registers in the IP thereby simplifying the IP interface is a practical design decision as far as the authors believe.

Performance of the circuit architecture in Figure 4 is evaluated in ASIC and FPGA. Table 5 and Table 6 summarizes the performances in ASIC and FPGA, respectively. The

Table 5: Performance of SAEB in ASIC.

	Standard Cell Library	Target	Circuit Area [GE]	Max. Freq. [MHz]	Latency [Cycles]
[BBM16]	*STMicroelectronics* 90-nm CMOS	CLOC	4,310	—	—
		SILC	4,220	—	—
		AES-OTR	6,770	—	—
This work	*NanGate* 45-nm CMOS	SAEB	3,502	122.0	231
		• AES	2,679	126.4	231
		• diff	823	—	—

performance is shown in three ways namely SAEB, AES, and diff. SAEB and AES correspond to the area of the whole and AES-only circuits, respectively. diff is a difference between SAEB and AES and represents an extra circuit area needed for the mode of operation.

Firstly, the performance in ASIC shown in Table 5 is discussed. *NanGate* 45-nm CMOS standard cell library is used for the evaluation [Nan]. SAEB is implemented with 3,502 [GE]: 2,679 [GE] for AES and extra 823 [GE] for the mode. The result shows that the overhead for the mode is small compared to the circuit area for AES. Note that even smaller AES implementations are possible at the cost of speed. For example, the design by Jean et al. [JMPS17] achieves 1,982 [GE] in the same cell library. The extra cost for SAEB is small even compared with the state-of-the-art compact AES implementations. The performance is compared with conventional implementations of CLOC, SILC, and AES-OTR by Banik et al. [BBM16]. In the previous work, CLOC, SILC, AES-OTR are implemented with $4,310$, $4,220$, and $6,770$ [GE], respectively. The circuit area of $3,502$ [GE] achieved by SAEB is smaller than any of them. The circuit areas needed CLOC, SILC, AES-OTR excluding a blockcipher implementation are $1,530$, $1,470$, and $4,220$ [GE], respectively. Again, 823 [GE] achieved by SAEB (diff in Table 5) is the smallest[5].

Secondly, the performance in FPGA shown in Table 6 is discussed. *Xilinx Virtex-7* and *Intel Cyclone V* are used as target devices. The table also shows performances in conventional works [CIMN17, GMU] for comparison. Chakraborti et al. evaluated hardware implementation of COFB [CIMN17] and showed that a circuit architecture supporting encryption only can be realized with $1,456$ [LUTs] and 722 [FFs] in *Virtex-7* FPGA (xc7vx330t). The circuit area of SAEB (348 [LUTs] and 242 FFs) is roughly 1/4 of the COFB implementation. This is not a fair comparison because Chakraborti et al. uses an AES implementation with 128-bit datapath. However, even a light-weight hardware implementation of COFB, designed with the same strategy as that in Figure 4, should be larger than that of SAEB. That is because (i) COFB requires extra 64-bit register for managing tweaks Δ in addition to a 128-bit state and (ii) a combinatorial circuit for updating the tweak is also needed. Table 6 also shows some data from the ATHENa database [GMU] in which hardware performances of the candidates for the CAESAR competition are registered. Among numerous results in the database, the smallest three implementations evaluated for *Virtex-7*, at the time of writing, are shown in the table. SAEB is smaller than ACORN (566 [LUTs]) which is the smallest in the database. This is a remarkable result considering that ACORN is a stream-cipher based dedicated design. Chakraborti et al. compared the aforementioned COFB implementation with ones in the ATHENa database. They conclude that COFB can be smaller than other block-cipher based candidates. The same is true for SAEB because SAEB can be smaller than COFB as discussed above.

[5]Banik et al. implemented two sets of implementations namely "Aggressive" and "Conservative". In this paper, "Conservative" is used for the evaluation because the "Aggressive" implementations expect some input restrictions e.g., AD should be empty.

Table 6: Performance of SAEB in FPGA.

	Platform	Target	Look-up Table [LUTs/ALMs]	Flip-flop [FFs]	Max. Freq. [MHz]	Latency [Cycles]
[CIMN17]	*Xilinx Virtex-7* `xc7vx330t`	`COFB`	1,456	722	264.2	—
[GMU]	*Xilinx Virtex-7*	`ACORN`	566	—	466.0	—
	`xc7vx485t`	`JAMBU`	1,051	—	491.0	—
	`ffg1761-3`	`ASCON`	1,557	—	444.0	—
This work	*Xilinx Virtex-7*	`SAEB`	348	242	145.9	231
	`xc7vx330t`	• `AES`	304	218	144.2	231
	`ffg1157-1`	• `diff`	44	24	—	—
This work	*Intel Cyclone V*	`SAEB`	299	410	83.3	231
	`5CEBA2F17C8`	• `AES`	264	283	87.8	231
		• `diff`	35	127	—	—

8 Conclusion

In this paper, a new lightweight blockcipher-based AEAD mode SAEB is proposed. The five good properties for blockcipher-based lightweight AEAD is discussed firstly. With the minimum state size and online properties, the size of working memory can be reduced. The inverse-free and XOR-only properties contribute to small program/circuit footprint. Efficient handling of static AD enables efficient computation based on precomputation. SAEB is the first mode that satisfy all the properties. SAEB is proved to be secure up to the birthday bound. Performance of SAEB is evaluated in software and hardware platforms. As a result, SAEB outperforms conventional algorithms in various performance metrics for lightweight cryptography.

Acknowledgments

We thank Megha Agrawal for pointing out typos in Figure 2.

References

[ADMA15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In *FSE 2015*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 411–436. Springer, 2015.

[BBM16] Subhadeep Banik, Andrey Bogdanov, and Kazuhiko Minematsu. Low-area hardware implementations of CLOC, SILC and AES-OTR. In *HOST 2016*, pages 71–74. IEEE Computer Society, 2016.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 208–225. Springer, 2012.

[BDPA08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Indifferentiability of the Sponge Construction. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 181–197. Springer, 2008.

[BDPA12a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *SAC 2011*, volume 7118 of *LNCS*, pages 320–337. Springer, 2012.

[BDPA12b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. In *DIAC 2012*, 2012.

[BJK+16] 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 *CRYPTO 2016*, volume 9815 of *LNCS*, pages 123–153. Springer, 2016.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[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 *CHES 2017*, volume 10529 of *LNCS*, pages 321–345. Springer, 2017.

[BR06] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, 2006.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[CAE] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html.

[CDH+12] Dong H. Chang, Morris J. Dworkin, Seokhie Hong, John M. Kelsey, and Mridul Nandi. A Keyed Sponge Construction with Pseudorandomness in the Standard Model, 2012.

[CIMN17] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-Based Authenticated Encryption: How Small Can We Go? In *CHES 2017*, volume 10529 of *LNCS*, pages 277–298. Springer, 2017.

[Dan] Quynh H. Dang. The Keyed-Hash Message Authentication Code (HMAC). FIPS PUB 198-1 (2008).

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 336–354. Springer, 2012.

[DMA17] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-State Keyed Duplex with Built-In Multi-user Support. In *ASIACRYPT 2017*, volume 10625 of *LNCS*, pages 606–637. Springer, 2017.

[Dwoa] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D (2007).

[Dwob] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. NIST Special Publication 800-38A (2001).

[EM91] Shimon Even and Yishay Mansour. A Construction of a Cioher From a Single Pseudorandom Permutation. In *ASIACRYPT '91*, volume 739 of *LNCS*, pages 210–224. Springer, 1991.

[EM97] Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *J. Cryptology*, 10(3):151–162, 1997.

[GL15] Shay Gueron and Yehuda Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In *ACM CCS 2015*, pages 109–119. ACM, 2015.

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

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED Block Cipher. In *CHES 2011*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.

[IMGM14] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, and Sumio Morioka. CLOC: Authenticated Encryption for Short Input. In *FSE 2014*, volume 8540 of *LNCS*, pages 149–167. Springer, 2014.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond 2 c/2 Security in Sponge-Based Authenticated Encryption Modes. In *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014.

[JMPS17] Jérémy Jean, Amir Moradi, Thomas Peyrin, and Pascal Sasdrich. Bit-Sliding: A Generic Technique for Bit-Serial Implementations of SPN-based Primitives - Applications to AES, PRESENT and SKINNY. In *CHES 2017*, LNCS, pages 687–707. Springer, 2017.

[KR11] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In *FSE 2011*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011.

[Min14] Kazuhiko Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 275–292. Springer, 2014.

[MMH+14] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers. In *SAC 2014*, volume 8781 of *LNCS*, pages 306–323. Springer, 2014.

[MPL+11] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.

[MV04] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004.

[Nan] NanGate Open Cell Library. http://www.nangate.com/.

[NRS14] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering Generic Composition. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 257–274. Springer, 2014.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In *CCS 2001*, pages 196–205. ACM, 2001.

[Ren] Renesas Electronics Corporation. RL78 Family. https://www.renesas.com/en-us/products/microcontrollers-microprocessors/rl78.html.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In *CCS 2002*, pages 98–107. ACM, 2002.

[Rog04] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, 2004.

[RS06] Phillip Rogaway and Thomas Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, 2006.

[SIH$^+$11] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An Ultra-Lightweight Blockcipher. In *CHES 2011*, volume 6917 of *LNCS*, pages 342–357. Springer, 2011.

[SMMK13] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In *SAC 2012*, volume 7707 of *LNCS*, pages 339–354. Springer, 2013.

[SSA$^+$07] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *FSE 2007*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007.

[WH] Hongjun Wu and Tao Huang. The JAMBU Lightweight Authentication Encryption Mode (v2.1). Submitted to the CAESAR competition (September 2016).

A The Detail Analysis of the Case (B) in forge

Fix $\alpha \in [q_{\mathcal{E}}]$ and $\beta \in [q_{\mathcal{D}}]$ such that the β-th decryption query is made after the α-th encryption query, and assume that $\mathsf{forge} = \mathbf{false}$. Then we show that if the case (B) occurs due to the α-th encryption and β-th decryption queries, i.e.,

$$\exists \alpha, \beta, i \text{ s.t. } sm^{\alpha}[i] = sc^{\beta}[m_\beta + 1],$$

then

$$\begin{aligned} \exists j \in \{0, \ldots, \min\{i-1, m_\beta\}\} \text{ s.t. } & sm^{\alpha}[i-(j+1)] \neq sc^{\beta}[m_\beta + 1 - (j+1)] \text{ and} \\ & sm^{\alpha}[i-j] = sc^{\beta}[m_\beta + 1 - j]. \end{aligned}$$

Let $(A^{\mathcal{E},\alpha}, C^{\mathcal{E},\alpha})$ (resp., $(A^{\mathcal{E},\beta}, C^{\mathcal{D},\beta})$) be AD and the ciphertext defined at the α-th encryption query (resp., the β-th decryption query). Let $sm[a,b] := sm[a] \| sm[a+1] \| \cdots \| sm[b]$ with $a \leq b$, and $sc[a,b] := sc[a] \| sc[a+1] \| \cdots \| sc[b]$. Then the following cases are considered.

- If $i \neq m_\alpha + 1$, then by the constant xor $\oplus \mathbf{1}_c / \oplus \mathbf{2}_c$ at the last block in `Core*.Dec`,
$$\begin{aligned} & sm^\alpha[i] = sc^\beta[m_\beta + 1] \\ & \quad \Rightarrow R(sm^\alpha[i-1]) \oplus R(sc^\beta[m_\beta]) = (M^\alpha_{i-1} \| const^\alpha[i]) \oplus (M^\alpha_{m_\beta} \| const^\beta[m_\beta + 1]) \neq 0, \end{aligned}$$
where $const^\alpha[i] = \mathbf{0}_c$, and $const^\beta[m_\beta + 1] = \mathbf{1}_c$ or $\mathbf{2}_c$. Thus, $sm^\alpha[i-1] \neq sc^\beta[m_\beta]$ (and $sm^\alpha[i] = sc^\beta[m_\beta + 1]$).
- If $i = m_\alpha + 1$ and $m_\alpha = m_\beta$, then
  - if $C^{\mathcal{E},\alpha} \neq C^{\mathcal{D},\beta}$ and $\exists a$ s.t. $M^\alpha_a \neq M^\beta_a$, then
$$\begin{aligned} \exists j \in \{0, \ldots, m_\alpha\} \text{ s.t. } & sm^\alpha[m_\alpha + 1 - (j+1)] \neq sc^\beta[m_\beta + 1 - (j+1)] \text{ and} \\ & sm^\alpha[m_\alpha + 1 - j] = sc^\beta[m_\beta + 1 - j]. \end{aligned}$$
  - if $C^{\mathcal{E},\alpha} \neq C^{\mathcal{D},\beta}$ and $\forall a \in [m_\alpha] : M^\alpha_a = M^\beta_a$, then $const^\alpha[m+1] \neq const^\beta[m+1]$, and thus
$$sm^\alpha[m_\alpha + 1] = sc^\beta[m_\alpha + 1] \wedge sm^\alpha[m_\alpha] \neq sc^\beta[m_\alpha].$$
  - if $C^{\mathcal{E},\alpha} = C^{\mathcal{D},\beta}$, then $(A^{\mathcal{E},\alpha}, N^{\mathcal{E},\alpha}) \neq (A^{\mathcal{E},\beta}, N^{\mathcal{D},\beta})$ is satisfied.
    If $A^{\mathcal{E},\alpha} \neq A^{\mathcal{D},\beta}$, then by $\mathsf{coll}_{\mathsf{sa}} = \mathbf{false}$ and the constant xor $\oplus \mathbf{1}_c / \oplus \mathbf{2}_c$, $sm^\alpha[0] \neq sc^\beta[0]$ is satisfied.
    If $A^{\mathcal{E},\alpha} = A^{\mathcal{D},\beta}$, i.e., $N^{\mathcal{E},\alpha} \neq N^{\mathcal{D},\beta}$, then $sm^\alpha[1] \neq sc^\beta[1]$ is satisfied.
    Hence,
$$\begin{aligned} \exists j \in \{0, \ldots, m_\alpha\} \text{ s.t. } & sm^\alpha[m_\alpha + 1 - (j+1)] \neq sc^\beta[m_\beta + 1 - (j+1)] \text{ and} \\ & sm^\alpha[m_\alpha + 1 - j] = sc^\beta[m_\beta + 1 - j]. \end{aligned}$$
- If $i = m_\alpha + 1$, $m_\alpha < m_\beta$ and $sm^\alpha[1, m_\alpha] \neq sc^\beta[m_\beta - m_\alpha + 1, m_\beta]$, then
$$\begin{aligned} \exists j \in \{1, \ldots, m_\alpha\} \text{ s.t. } & sm^\alpha[m_\alpha + 1 - (j+1)] \neq sc^\beta[m_\beta + 1 - (j+1)] \text{ and} \\ & sm^\alpha[m_\alpha + 1 - j] = sc^\beta[m_\beta + 1 - j]. \end{aligned}$$
- If $i_\alpha = m_\alpha + 1$, $m_\alpha < m_\beta$ and $sm^\alpha[1, m_\alpha] = sc^\beta[m_\beta - m_\alpha + 1, m_\beta]$, then
$$\begin{aligned} & sm^\alpha[1] = sc^\beta[m_\beta - m_\alpha + 1] \\ & \quad \Rightarrow R(sm^\alpha[0]) \oplus R(sc^\beta[m_\beta - m_\alpha]) = (N^{\mathcal{E},\alpha} \| \mathbf{3}_{c_2}) \oplus (M^\alpha_{j-1} \| \mathbf{0}_c) \neq 0. \end{aligned}$$
Thus, $sm^\alpha[0] \neq sc^\beta[m_\beta - m_\alpha]$ (and $sm^\alpha[1] = sc^\beta[m_\beta - m_\alpha + 1]$).
- If $i_\alpha = m_\alpha + 1$ and $m_\alpha > m_\beta$, then by the same analyses as the above two cases,
$$sm^\alpha[m_\alpha - m_\beta] \neq sc^\beta[0] \wedge sm^\alpha[m_\alpha - m_\beta + 1] = sc^\beta[0].$$

Hence, if the case (B) occurs, then
$$\begin{aligned} \exists j \in \{0, \ldots, \min\{i-1, m_\beta\}\} \text{ s.t. } & sm^\alpha[i - (j+1)] \neq sc^\beta[m_\beta + 1 - (j+1)] \text{ and} \\ & sm^\alpha[i - j] = sc^\beta[m_\beta + 1 - j]. \end{aligned}$$