JHAE: A Novel Permutation-Based Authenticated Encryption Mode Based on the Hash Mode JH

Javad Alizadeh[1], Mohammad Reza Aref[1], Nasour Bagheri[2]

[1] Information Systems and Security Lab. (ISSL), Electrical Eng. Department, Sharif University of Technology, Tehran, Iran, alizadja@gmail, aref@sharif.edu
[2] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran, NBagheri@srttu.edu

Abstract. In this paper JHAE, an authenticated encryption (AE) mode, was presented based on the JH hash mode. JHAE is an on-line and single-pass dedicated AE mode based on permutation that supports optional associated data (AD). It was proved that this mode, based on ideal permutation, achieved privacy and integrity up to $O(2^{n/2})$ queries where the length of the used permutation was $2n$. To decrypt, JHAE did not require the inverse of its underlying permutation and therefore saved area space. JHAE has been used by Artemia, one of the CAESAR candidates.

Keywords: Authenticated Encryption, Provable Security, Privacy, Integrity, CAESAR

1 Introduction

Privacy and authentication are two main goals in information security. In many applications, these security parameters are established simultaneously. For example, in the widely used Transport Layer Security (TLS), a generic approach (the MAC-then-Encrypt approach [3]) is used. A cryptographic scheme that provides both privacy and authentication is called authenticated encryption (AE) scheme. The traditional approach for AE is the use of generic compositions. In this approach, two algorithms are used, one of which provides confidentiality and the other provides authenticity. However, this approach is not efficient for many applications, because it requires two different algorithms with two different keys as well as separate passes over the message [3]. Another approach for designing an AE is the use of a block cipher in a special mode, in which the block cipher is treated as a black box in the mode [26, 36, 39]. The most important problem of these modes is the necessity for a running the full round block cipher to process each message block which is time/resource-consuming.

Dedicated AE schemes resolve the problems of generic compositions and block cipher based modes. Designing a dedicated AE has recently received great attention in cryptography community, mostly driven by the NIST-funded CAESAR competition for AE [12]. ASC-1 [19], ALE [11], AEGIS [41], FIDES [10], CBEAM [37], and APE [2] are some dedicated AE schemes which were submitted before the CAESAR. A common approach for constructing a dedicated AE is to iterate a random permutation or random function in a special mode of operation. Therefore, there are two main stages in designing a new dedicated AE:

1. Designing a new dedicated mode (based on a random permutation or a random function)

2. Designing a new random permutation or random function to be used in the mode.

A general approach is to design a dedicated AE mode from a hash function mode. For example, duplex constructions [5], which were used in designing of the CAESAR candidates Ascon [13], ICEPOLE [34], KETJE [35], KEYAK [28], NORX [21], π-Cipher [15], PRIMATEs-GIBBON [16], PRIMATEs-HANUMAN [16], PRIMATEs-APE [16], PRØST-APE [17], and STRIBOB [38], are closely related to the sponge construction [6]. Other examples include FWPAE and FPAE modes [25] that were obtained from FWP [30] and FP [33] hash function modes, respectively. An important challenge in developing an AE mode from another mode (e.g. hash mode) is to prove its security to ensure transition to another application does not make any structural flaws.

Hash Modes. A hash function has two main components, a mode of operation, and a primitive which is iteratively used by the mode. For example the Merkle-Damgård construction [14, 27] was used in designing of many famous hash functions such as SHA-0 [31] and SHA-1 [32]. Some flaws in the construction (e.g. multi-collision attack [22]) leads to development of new hash constructions such as Wide-pipe [24], Sponge [6], JH [40], Grøstl [18], and FP [33]. The last four ones are permutation-based hash modes. JH and Grøstl were two finalists of the NIST SHA-3 hash function competition and Sponge was used by the hash function Keccak [8] which was the winner of the competition. A comparison of some hash function modes was presented in [33]. For the modes Sponge, Grøstl, JH, and FP the comparison was summarized in Table 1 where ϵ is a small fraction due to the preimage attack on JH presented in [9]. Some of the advantages of permutation-based hash modes were given as follows:

- The modes do not need any key schedule.
- Easy-to-invert permutations are usually efficient [33].

Table 1. Comparison of some permutation-based hash modes [33].

Mode	Mesg-blk (l)	Size of π (a)	Rate (l/a)	Indiff. bound lower	Indiff. bound upper	# of independent permutations	Reference
Sponge	n	$2n$	0.5	$n/2$	$n/2$	1	[7]
Grøstl	n	$2n$	0.5	$n/2$	n	2	[18]
JH	n	$2n$	0.5	$n/2$	$n(1-\epsilon)$	1	[29]
FP	n	$2n$	0.5	$n/2$	n	1	[33]

Contribution . In this paper JH hash function mode [40] is used to develop a new dedicated AE mode, called JHAE. The main reasons of using JH mode to design a new AE mode were given as follows:

- It was a permutation-based mode.

- Keccak (which used the Sponge construction), Grøstl, and JH were three finalists of the SHA-3 competition. In comparison by Grøstl, JH used only one permutation and in comparison by Sponge, it had better indifferentiability upper bound (See Table 1).
- Duplex constructions [5] and FPAE [25] were two AE modes based on the Sponge and FP hash function modes, respectively, and until now no AE mode is presented based on the JH hash function mode.
- The important researches on the JH hash mode had done in the duration of SHA-3 competition and shown that there was not any significant vulnerability in the mode.

JHAE is an on-line and single-pass dedicated AE mode that supports optional associated data (AD). Also, its security relies on using nonces. It was proved in this paper that the mode achieved privacy (indistinguishability under the chosen plaintext attack or IND-CPA) and integrity (integrity of ciphertext or INT-CTXT) up to $O(2^{n/2})$ queries, where the length of the used permutation was $2n$. In addition, it was demonstrated that the integrity bound of JHAE was reduced to the indifferentiability of JH hash mode, which is at least $O(2^{n/2})$.

JHAE in the CAESAR Competition. Artemia [20] is a family of the dedicated authenticated encryption scheme which was submitted to the CAESAR competition. Artemia is a sponge-based authenticated encryption scheme that uses the JHAE mode. Except Artemia, all of the sponge-based candidates of CAESAR used the duplex constructions [1]. Until now (in the duration of the CAESAR competition) no flaw has been reported for Artemia. A comparison between Artemia and other dedicated AE schemes which were submitted to the CAESAR competition was presented in [1]. With respect to [1], the comparison of Artrmia and other sponge-based candidates can be summarized as Table 2. The features of the schemes were inherited from their mode (e.g. the features of Artemia were inherited from JHAE).

Table 2. Comparison between Artemia and other sponge-based candidates of CAESAR [1]. n.n. means unnamed custom primitive.

Sponge-Based AE	Design	Primitive	Provable Security	Parallelizable	On-Line	Nonce Misuse Resistance	Inverse-Free	Refference
Artemia	JHAE	Artemia	Yes	No	Yes	No	Yes	[20]
Ascon	Duplex	Ascon	No	No	Yes	Yes	Yes	[13]
ICEPOLE	Duplex	n.n.	Yes	Yes	Yes	Yes	Yes	[34]
KETJE	Duplex	Keccak-f	Yes	No	Yes	No	Yes	[35]
KEYAK	Duplex	Keccak-f	Yes	Yes	Yes	No	Yes	[28]
NORX	Duplex	n.n.	No	Yes	Yes	No	Yes	[21]
π-Cipher	Duplex	n.n.	No	Yes	Yes	No	Yes	[15]
PRIMATEs-GIBBON	Duplex	PRIMATE	No	No	Yes	No	Yes	[16]
PRIMATEs-HANUMAN	Duplex	PRIMATE	No	No	Yes	No	Yes	[16]
PRIMATEs-APE	Duplex	PRIMATE	Yes	No	Yes	Yes	No	[16]
PRØST-APE	Duplex	PRØST	Yes	No	Yes	Yes	No	[17]
STRIBOB	Duplex	n.n	Yes	No	Yes	No	Yes	[20]

Organization. The paper is structured as follows: Section 2 gives a specification of JHAE encryption-authentication and decryption-verification. Security of JHAE is analyzed in Section 3. In this section, privacy and integrity of JHAE, are proved in the ideal permutation model and by reducing to the security of JH hash mode, respectively. In Section 4, the rationale behind of the JHAE design is briefly described. Finally conclusion is given in Section 5.

2 JHAE Authenticated Encryption Mode

In this section, JHAE mode, depicted in Fig 4, is described. JHAE is developed from JH hash function mode (Fig 3) [40] and iterates a fixed permutation $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$. It is a nonce-based, single-pass, and on-line dedicated AE mode that supports AD. To decrypt, JHAE does not require the inverse of its underlying permutation and therefore saved area space.

2.1 Encryption and Authentication

JHAE accepts an n-bit key K, an n-bit nonce N, a message M, an optional AD, A, and produces ciphertext C and authentication tag T. Pseudo-code of JHAE's encryption-authentication is depicted in Table 3. It is assumed that the input message, after padding, is a multiple of the block size (n). The last block of the original message is concatenated by the padding data as follows (See Figure 1):

1. The length of nonce (N) is appended to the end of the last block of message.
2. The length of the associated data (A) is appended to the end of the padded message in 1.
3. The length of the message (M) is appended to the end of the padded message in 2.
4. A bit '1' followed by a sequence of '0' is appended to the end of the padded message in 3 such that the padded message is a multiple of the block size n.

If there is the AD in the procedure, it is padded by a bit '1' followed by a sequence of '0' such that the padded AD would be a multiple of the block size n (See Figure 2). The padded AD is processed in a way which is similar to the process of the message block with an exception that ciphertext blocks (c_i), are not produced for the AD blocks.

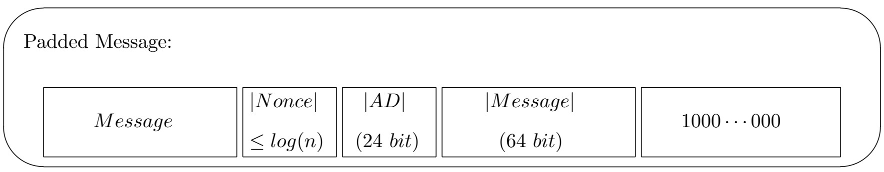

Fig. 1. Message padding in JHAE

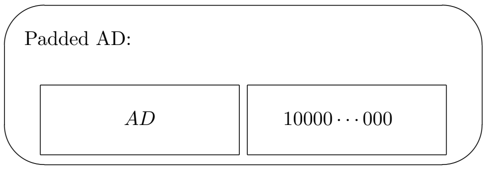

Fig. 2. AD padding in JHAE

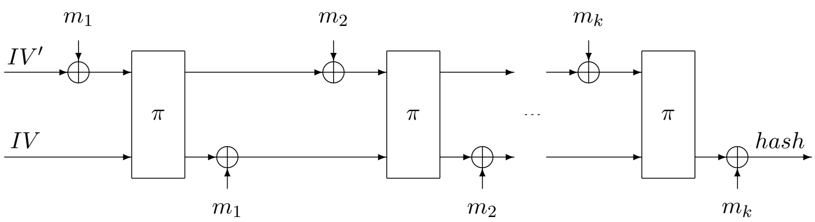

Fig. 3. JH hash mode [29]

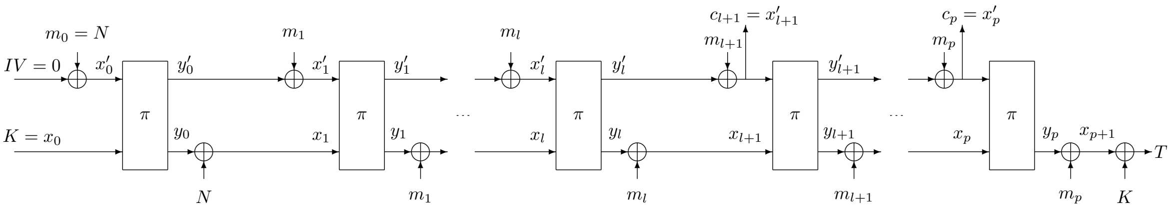

Fig. 4. JHAE mode of operation (encryption and authentication), where $pad(A) = m_1 \| m_2 \| \ldots \| m_l$ and $pad(M) = m_{l+1} \parallel m_{l+2} \parallel ... \parallel m_p$

Table 3. Encryption and authentication pseudo code of JHAE

Algorithm1. $JHAE - E^{\pi}(K, N, M, A)$

Input: Key K of n bits, Nonce N of n bits, Associated data A where $pad(A) = m_1 \| m_2 \| \ldots \| m_l$ and Message M where $pad(M) = m_{l+1} \parallel m_{l+2} \parallel ... \parallel m_p$
Output: Ciphertext C, Tag T
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p - 1$ do:
    $y'_i \parallel y_i = \pi(x'_i \parallel x_i)$;
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p = \pi(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)

2.2 Decryption and Verification

JHAE decryption-verification procedure, depicted in Table 4, accepts an n-bit key K, an n-bit nonce N, a ciphertext C, a tag T, an optional AD, A, and decrypts the ciphertext to get message M and tag T'. If $T' = T$, then it outputs M; else, it outputs $\perp$.

3 Security Proofs

In this section, security of JHAE is proved. First, game playing framework proposed by Bellare and Rogaway [4] is used and an upper bound is obtained for the advantage of an adversary that can distinguish the JHAE from a random oracle (IND-CPA) in the ideal permutation model. Then, it is proved that JHAE provides integrity (INT-CTXT) until JH hash mode is indifferentiable from a random oracle or tag can not be guessed. These proofs are followed in two subsections of privacy and integrity.

3.1 Privacy

In this section, privacy's security bound for JHAE based on ideal permutation π is provided.

Theorem 1. *JHAE based on an ideal permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$, is (t_A, σ, ϵ)-indistinguishable from an ideal AE based on a random function RO and ideal permutation π' with the same domain and range, for any t_A; then, $\epsilon \le \dfrac{\sigma(\sigma-1)}{2^{2n-1}} + \dfrac{\sigma^2}{2^{2n}} + \dfrac{\sigma^2}{2^n}$, where σ is the total number of blocks in queries to $JHAE$ encryption function (denoted by $JHAE - E$), π, and π^{-1}, by the adversary $\mathcal{A}$.*

Table 4. Decryption and verification pseudo code of JHAE

Algorithm2. $JHAE - D^{\pi}(K, N, C, T, A)$

Input: Key K of n bits, Nonce N of n bits, Associated Data A where $pad(A) = m_1 \| m_2 \| \ldots \| m_l$
, ciphertext $C = c_1 \parallel c_2 \parallel ... \parallel c_p$ and Tag T
Output: Message M or $\perp$
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
$x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_{l+p} = c_1 \parallel c_2 \parallel ... \parallel c_p$
for $i = 0$ to $l - 1$ do:
    $y'_i \parallel y_i = \pi(x'_i \parallel x_i)$;
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
for $i = l$ to $p - 1$ do:
    $y'_i \parallel y_i = \pi(x'_i \parallel x_i)$;
    $m_{i+1} = y'_i \oplus x'_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p = \pi(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$M = m_{l+1} \parallel m_{l+2} \parallel ... \parallel m_p$
$T' = x_{p+1} \oplus K$
if $T' = T$
  Return M
else
  Return $\perp$

Proof. To prove the above theorem, a game playing framework based on ten games of G_0 to G_9 is used where G_0 represents JHAE based on ideal permutation π , $JHAE-\pi,\pi^{-1}$, and G_9 represents a random oracle, RO, an ideal permutation π and its inverse π^{-1}. To determine the adversary's advantage on distinguishing JHAE from an ideal AE scheme, the adversary's advantage moving from a game to the next game is calculated.

Game G_0. This game shows the communication of $\mathcal{A}$ with $JHAE-\pi,\pi^{-1}$ (see Table 5). In this game, permutations π and π^{-1} are exactly the permutations that are used in the real JHAE mode. Hence:

$$Pr[\mathcal{A}^{G_0} \Rightarrow 1] = Pr[\mathcal{A}^{JHAE-E} \Rightarrow 1]$$

Game G_1. This game is identical to G_0 with an exception that the ideal permutation (π,π^{-1}) is chosen in a "lazy" manner, oracles O_2 and O_3 respectively (see Table 6). These oracles perfectly simulate two ideal permutations and, since it is assumed that π and π^{-1} in G_0 are ideal permutations, then the distribution of the returned values in G_0 and G_1 are identical. Therefore we have:

$$Pr[\mathcal{A}^{G_1} \Rightarrow 1] = Pr[\mathcal{A}^{G_0} \Rightarrow 1].$$

Game G_2. To generate G_2, a PRP-PRF switch [4] is done in G_1(see Table 7). This means that the ideal permutations O_2 and O_3 in G_1 are replaced with two random functions in G_2. Therefore, the only difference between G_2 and G_1 is oracles O_2 and O_3 (two ideal permutations are stimulated in G_1; but, two random functions are stimulated in G_2). Unlike the ideal permutation, it is possible to find a collision in a random function. Since in G_1, there is not collision, in G_2, There may be a collision in O_2 or O_3 and the adversary can differentiate G_2 from G_1. Hence, a collision is defined in G_2 as a bad event and denoted by bad_0. The distribution of the returned values by G_2 and G_1 are identical until bad_0 occurs. Suppose that the adversary can do at most σ_2 and σ_3 query for O_2 and O_3, respectively, and let $\sigma' = \sigma_2 + \sigma_3$; Then:

$$Pr[\mathcal{A}^{G_2} \Rightarrow 1] - Pr[\mathcal{A}^{G_1} \Rightarrow 1] = Pr[bad_0 \leftarrow true] = Pr[Collision\ in\ O_2\ or\ O_3\ in\ G2]$$

$$\leq \frac{\sigma_2(\sigma_2-1)}{2^{2n+1}} + \frac{\sigma_3(\sigma_3-1)}{2^{2n+1}} \leq \frac{\sigma'(\sigma'-1)}{2^{2n+1}} \leq \frac{\sigma(\sigma-1)}{2^{2n+1}}.$$

Game G_3. In G_3, oracle O_1 does not pass any query to the oracle O_2; but, it exactly simulates the behavior of oracle O_2(see G_3 in Table 8). Thus, the distribution of the returned values by G_3 and G_2 are identical from the adversary's view:

$$Pr[\mathcal{A}^{G_3} \Rightarrow 1] = Pr[\mathcal{A}^{G_2} \Rightarrow 1].$$

Game G_4. In G_4 (see Table 9) the purpose is to push the behavior of O_1 one step towards the random oracle. Hence, the queries that are included into O_2 by O_1 and those that are directly queried by the adversary of O_2 or O_3 are separated. In this game, if an intermediate query generated by O_1, that is expected to be queried to O_2, has a record on the part of O_2 not included by O_1, it is considered a bad event and denoted by bad_1. However, the distribution of responses of queries to O_2 and O_3 remains identical to G_3. Hence, it can be stated that G_3 and G_4 are identical until bad_1 occurs in G_4. Assuming that the adversary can do at most σ_1 query to O_1 and σ' query to O_2 or O_3, the adversary's advantage from G_3 to G_4 is bounded as follows:

$$Pr[\mathcal{A}^{G_4} \Rightarrow 1] - Pr[\mathcal{A}^{G_3} \Rightarrow 1] = Pr[bad_1 \leftarrow true] \leq \frac{\sigma'(\sigma_1)}{2^{2n}} \leq \frac{\sigma^2}{2^{2n}}.$$

Game G_5. In G_5 (see Table 10), the responses of O_2 or O_3 are not compatible with those of O_1. In G_5, the purpose is to push the behaviour of O_2 and O_3 one step towards the ideal permutations that are independent from RO. For this reason, two auxiliary tables are generated to keep the input and output of the intermediate tentative queries to O_2 generated by O_1 which are denoted by W and Y, respectively. The aim of this game is to not return any record that has been included in O_2 by O_1 when the adversary is directly queried to O_2 or O_3. Hence, in this game, if a query to O_2 or O_3 has a record in W and Y, respectively, it is considered a bad event and denoted by bad_2. More precisely, on query to O_1, when it generates a local tentative fresh query w_i to O_2 and generates y_i as a response, then w_i is stored in W and y_i is stored in Y. However, distribution of the responses to queries to O_1 remains identical to G_4. Hence, it can be stated that G_4 and G_5 are identical until bad_2 occurs in G_4. To bound the probability of bad_2, suppose that w_j is the j-th block that is queried to O_1 and y_j is the response of O_1 to the query where $1 \leq j \leq \sigma_1$, v_i is the i-th query to O_2 where $1 \leq i \leq \sigma_2$, and z_l is the l-th query to O_3 where $1 \leq l \leq \sigma_3$. Then:

$$Pr[bad_2 \leftarrow true] = \sum_{i=1}^{\sigma_2}\sum_{j=1}^{\sigma_1} Pr[v_i = w_j] + \sum_{l=1}^{\sigma_3}\sum_{j=1}^{\sigma_1} Pr[z_l = y_j] \leqslant \frac{\sigma_2\sigma_1}{2^n} + \frac{\sigma_3\sigma_1}{2^n}.$$

It must be noted that, in the above calculations, the fact that, given the response of a query to O_1, the adversary can determine half of the bits of each $w_j \in W$ and $y_i \in Y$ is considered. Hence, the adversary's advantage from G_4 to G_5 is bounded as follows:

$$Pr[\mathcal{A}^{G_5} \Rightarrow 1] - Pr[\mathcal{A}^{G_4} \Rightarrow 1] \leq \frac{\sigma_1 \times (\sigma_2 + \sigma_3)}{2^n} \leqslant \frac{\sigma^2}{2^n}.$$

Game G_6. G_6 (see Table 11) is identical to G_5 with an exception that O_1 does not keep the history of the intermediate queries. However, this modification has no impact on the distribution of the returned values to the adversary, if there is no bad event in neither of the games. Hence, in the adversary's view, for queries to O_1, distributions of the returned values in G_5 and G_6 are identical as far as there is not an intermediate collision in G_5. On the other hand, the distribution of responses to queries to O_2 and O_3 remains identical to G_5. Hence, the adversary's advantage from G_5 to G_6 is bounded as follows:

$$Pr[\mathcal{A}^{G_6} \Rightarrow 1] - Pr[\mathcal{A}^{G_5} \Rightarrow 1] \leq \frac{\sigma_1 \times (\sigma_1 - 1)}{2^{2n}} \leq \frac{\sigma \times (\sigma - 1)}{2^{2n}}.$$

Game G_7. In Game G_7 (see Table 12), the blocks of ciphertext and tag value are generated randomly. However, it has no impact of the distribution of the returned values to the adversary. Hence, distributions of the returned values in G_6 and G_7 are identical:

$$Pr[\mathcal{A}^{G_7} \Rightarrow 1] = Pr[\mathcal{A}^{G_6} \Rightarrow 1].$$

Game G_8. In Game G_8 (see Table 12), a PRF-PRP switch [4] is run; i.e. the ideal random functions O_2 and O_3 in G_7 are replaced with a random permutation and its inverse in G_8. Therefore, the only difference between G_7 and G_8 is oracles O_2 and O_3. Thus, the distribution of the returned values by G_7 and G_8 are identical until O_2 or O_3 has a collision in G_7. Hence, the adversary's advantage from G_7 to G_8 is bounded as follows:

$$Pr[\mathcal{A}^{G_8} \Rightarrow 1] - Pr[\mathcal{A}^{G_7} \Rightarrow 1] = Pr[Collision\ in\ O_2\ or\ O_3\ in\ G_7]$$
$$\leq \frac{\sigma_2(\sigma_2 - 1)}{2^{2n+1}} + \frac{\sigma_3(\sigma_3 - 1)}{2^{2n+1}} \leq \frac{\sigma'(\sigma' - 1)}{2^{2n+1}} \leq \frac{\sigma(\sigma - 1)}{2^{2n+1}}.$$

Game G_9. In G_8 for each message/AD block, an appropriate (regarding the length) random value is selected as cipher text and similarly a random value is selected as the tag value. Next, these random values are concatenated and returned to the adversary. However, in G_9 (see Table 13) on query to O_1, a random string of the length of the desired cipher and tag is selected and returned to the adversary. However, this modification from G_8 to G_9 has no impact on the distribution of the returned values to the adversary. Hence:

$$Pr[\mathcal{A}^{G_9} \Rightarrow 1] = Pr[\mathcal{A}^{G_8} \Rightarrow 1].$$

On the other hand, G_8 perfectly simulates RO, π, π^{-1}. Then:

$$Pr[\mathcal{A}^{RO,\pi,\pi^{-1}} \Rightarrow 1] = Pr[\mathcal{A}^{G_9} \Rightarrow 1].$$

Finally, using the fundamental lemma of game playing [4], the following can be stated:

$$\begin{aligned}
Adv_{JHAE}^{Privacy}(\mathcal{A}) &= Pr[\mathcal{A}^{JHAE-E,\pi,\pi^{-1}} \Rightarrow 1] - Pr[\mathcal{A}^{RO,\pi,\pi^{-1}} \Rightarrow 1] \\
&= Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_9} \Rightarrow 1] \\
&= (Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_1} \Rightarrow 1]) \\
&\quad +(Pr[\mathcal{A}^{G_1} \Rightarrow 1] - Pr[\mathcal{A}^{G_2} \Rightarrow 1]) \\
&\quad +(Pr[\mathcal{A}^{G_2} \Rightarrow 1] - Pr[\mathcal{A}^{G_3} \Rightarrow 1]) \\
&\quad +(Pr[\mathcal{A}^{G_3} \Rightarrow 1] - Pr[\mathcal{A}^{G_4} \Rightarrow 1]) \\
&\quad +(Pr[\mathcal{A}^{G_4} \Rightarrow 1] - Pr[\mathcal{A}^{G_5} \Rightarrow 1]) \\
&\quad +(Pr[\mathcal{A}^{G_5} \Rightarrow 1] - Pr[\mathcal{A}^{G_6} \Rightarrow 1])
\end{aligned}$$

$$
\begin{aligned}
&+(Pr[\mathcal{A}^{G_6} \Rightarrow 1] - Pr[\mathcal{A}^{G_7} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_7} \Rightarrow 1] - Pr[\mathcal{A}^{G_8} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_8} \Rightarrow 1] - Pr[\mathcal{A}^{G_9} \Rightarrow 1]) \\
\leq 0 + \frac{\sigma(\sigma-1)}{2^{2n+1}} + 0 + \frac{\sigma^2}{2^{2n}} + \frac{\sigma^2}{2^n} + \frac{\sigma(\sigma-1)}{2^{2n}} + 0 + \frac{\sigma(\sigma-1)}{2^{2n+1}} + 0 \\
&\leq \frac{\sigma(\sigma-1)}{2^{2n-1}} + \frac{\sigma^2}{2^{2n}} + \frac{\sigma^2}{2^n}.
\end{aligned}
$$

3.2 Integrity

In this section, integrity of ciphertext (INT-CTXT) of JHAE is proved. The INT-CTXT security bound of a permutation based AE scheme is defined as the maximum advantage of any adversary to produce a valid triple $(N, A\|C, T)$ (e.g. a forgery for the AE scheme) without directly querying to the scheme. To forge an AE scheme, the adversary can query to $AE - E$ (encryption and authentication), $AE - D$ (decryption and verification), and π or π^{-1}. Thus, two phases can be considered for any forgery attempt as follows:

1. **Data gathering:** The adversary gathers some valid triples such as $S = (N_i, (A\|C)_i, T_i); 1 \leq i \leq q$ by at most q queries to $AE - E$, π or π^{-1}.
2. **Execution:** The adversary produces a new triple $(N, A\|C, T)$ such that $(N, A\|C, T) \notin S$ is accepted by $AE - D$ as a valid triple.

In this section, it is shown that the advantage of any adversary that makes a reasonable number of queries to $JHAE - E$, π, and π^{-1} is negligible in the forgery attack against $JHAE$.

Theorem 2. *For any adversary $\mathcal{A}$ that makes total σ block queries to $JHAE - E$, π, or π^{-1}, JHAE based on an ideal permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$, is (t_A, σ, ϵ)-unforgeable, for any t_A, where $\epsilon \leq \frac{\sigma^2}{2^n} + \frac{q}{2^n}$.*

Proof. Suppose that $\mathcal{A}$ is an adversary that tries to forge JHAE. $\mathcal{A}$ should query at the first to JHAE, q times, and produce a list $S = \{(N_i, (A\|C)_i, T_i); 1 \leq i \leq q\}$. Next, $\mathcal{A}$ produces a new $(N, A\|C, T) \notin S$ such that $JHAE - D(N, A\|C, T) \neq \perp$ as its forged triple. All of the possible cases for the new valid $(N, A\|C, T)$ are as follows (cases 001 to 111).

1. **Case 001.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\exists (N_i, (A\|C)_i, T_i) \in S : N = N_i, A\|C = (A\|C)_i, T \neq T_i$, for $0 \leq i \leq q$.
2. **Case 010.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\exists (N_i, (A\|C)_i, T_i) \in S : N = N_i, A\|C \neq (A\|C)_i, T = T_i$, for $0 \leq i \leq q$.
3. **Case 011.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\forall (N_i, (A\|C)_i, T_i) \in S : A\|C \neq (A\|C)_i, T \neq T_i$, for $0 \leq i \leq q$ and $\exists (N_i, (A\|C)_i, T_i) \in S : N = N_i, A\|C \neq (A\|C)_i, T \neq T_i$.
4. **Case 100.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\exists (N_i, (A\|C)_i, T_i) \in S : N \neq N_i, A\|C = (A\|C)_i, T = T_i$, for $0 \leq i \leq q$.
5. **Case 101.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\exists (N_i, (A\|C)_i, T_i) \in S : N \neq N_i, A\|C = (A\|C)_i, T \neq T_i$, for $0 \leq i \leq q$.

6. **Case 110.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\exists(N_i, (A\|C)_i, T_i) \in S : N \neq N_i, A\|C \neq (A\|C)_i, T = T_i$, for $0 \leq i \leq q$.
7. **Case 111.** Adversary generates a valid $(N, A\|C, T) \notin S$ such that $\forall(N_i, (A\|C)_i, T_i) \in S : N \neq N_i, A\|C \neq (A\|C)_i, T \neq T_i$, for $0 \leq i \leq q$.

Hence, the adversary's advantage can be upper bound to forge JHAE as follows:

$$\begin{aligned} Pr[\mathcal{A}^{INT}_{JHAE} \Rightarrow 1] = Pr[Case\ 001] + Pr[Case\ 010] + Pr[Case\ 011] \\ + Pr[Case\ 100] + Pr[Case\ 101] + Pr[Case\ 110] + Pr[Case\ 111]. \end{aligned} \tag{1}$$

To determine an upper bound for this advantage, the mentioned cases are categorized as three distinct sets as follows and the adversary's advantage in producing a successful forgery for each set is determined.

Set 1: Set 1 includes any case that could not be used to successfully forge JHAE. More precisely, any triple that matches case 001 can not be used to forge JHAE. The reason comes from the fact that, for JHAE for a valid triple, if $A\|C = (A\|C)_i$ and $N = N_i$ then $T = T_i$. Therefore:

$$Pr[Case\ 001] = 0.$$

Set 2: Set 2 includes any case that can be directly used to differentiate JH hash mode from a random oracle. To determine these cases, JH hash mode in Fig 3 is considered. Since $T = T_i$ (for $1 \leq i \leq q$) implies $(x_{p+1})_i = (x_{p+1})$, and $(x_{p+1})_i$ and (x_{p+1}) are hash outputs in JH hash mode, then cases 010, 100, and 110 in the forgery attempt of JHAE lead to collisions in JH hash mode. In other words, if cases 010, 100, and 110 occur in the forgery attempt of JHAE, a collision can be found in the JH hash mode and therefore the mode can be dierentiated from a random oracle. Since the bound of the indifferentiability of JH has been proved to be $\frac{\sigma^2}{2^n}$ [29], then:

$$Pr[Case\ 010] + Pr[Case\ 100] + Pr[Case\ 110] \leq \frac{\sigma^2}{2^n}.$$

Set 3: This set includes cases that force the adversary to guess the tag. More precisely, in cases 011, 101, and 111, the adversary finds a new valid $(N, A\|C, T)$ such that $\forall(N_i, (A\|C)_i, T_i) \in S : N \neq N_i$ or $A\|C \neq (A\|C)_i$. On the other hand, given such a pair of N and $A\|C$, distribution of the valid tag would be uniformly distributed over $\{0, 1\}^n$. Hence, at each attempt, the adversary's advantage in generating a valid tag would be 2^{-n}. So:

$$Pr[Case\ 101] + Pr[Case\ 011] + Pr[Case\ 111] \leq \frac{q}{2^n}$$

Finally, using Equation 1:

$$Pr[\mathcal{A}^{INT}_{JHAE} \Rightarrow 1] \leq \frac{\sigma^2}{2^n} + \frac{q}{2^n}$$

4 Design Rationale

In this section, design rationale of JHAE, is described briefly.

Structure. The structure of JHAE is based on the JH hash function mode. The rational of using JH mode was mentioned in Section 1.

Padding. In the padding rule of JHAE, the length of nonce, AD, and message were used. The main rational of the rule is domain separation between nonce, AD, and message.

Final Key Addition. With respect to Figure 4, the final tag was computed as $x_{p+1} \oplus K$. Since JHAE didn't use explicit finalization, this key addition is required to prevent the length extension attacks.

5 Conclusion

In this paper, JHAE, a new dedicated permutation-based *AE* mode, was introduced. JHAE is an on-line and single-pass dedicated *AE* mode which did not require the inverse of its underlying permutation to decrypt and therefore saved area space. JHAE was used by Artemia, one of the CAESAR candidates.

In the ideal permutation model, it was proved that JHAE provided IND-CPA and INT-CTXT up to $q = O(2^{n/2})$. This is the first nontrivial security bound for JHAE. On the other hand, the best-known attack on JHAE has a complexity up to $q = O(2^n)$. Therefore, in particular there remains a gap between the best-known attack and the security bound of JHAE.

In a recent work, Jovanovic et. al. [23] showed that sponge based constructions for authenticated encryption, namely JHAE, can achieve the significantly higher bound of $2^{c/2}$, where c is their capacity. (Note that the capacity of JHAE, is n). For a future work, the security bound of JHAE can be improved using the security model introduced in [23].

References

1. F. Abed, C. Forler, and S. Lucks. Classification of the CAESAR Candidates. IACR Cryptology ePrint Archive, 2014.
2. E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda. APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. Preproceedings of Fast Software Encryption (FSE 2014), 2014. To Appear.
3. M. Bellare and C. Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.
4. M. Bellare and P. Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.
5. G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In *Selected Areas in Cryptography SAC*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.
6. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge Functions. ECRYPT hash workshop, 2007.
7. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology–EUROCRYPT 2008*, pages 181–197. Springer, 2008.
8. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak sponge function family main document. Submission to NIST, 2009.
9. R. Bhattacharyya, A. Mandal, and M. Nandi. Security analysis of the mode of JH hash function. In *Fast Software Encryption*, pages 168–191. Springer, 2010.
10. B. Bilgin, A. Bogdanov, M. Knezevic, F. Mendel, and Q. Wang. FIDES: Lightweight Authenticated Cipher with Side-Channel Resistance for Constrained Hardware. In *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013.
11. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. ALE: AES-based lightweight authenticated encryption. Preproceedings of Fast Software Encryption (FSE 2013), 2013. To Appear.
12. CAESAR. CAESAR: Competition for Authenticated Encryption: Security, Applicability,and Robustness, 2013. `http://competitions.cr.yp.to/caesar.html`.
13. F. M. Christoph Dobraunig, Maria Eichlseder and M. Schlffer. Ascon v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
14. I. B. Damgård. A design principle for hash functions. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 416–427. Springer, 1990.
15. S. S. H. J. M. E.-H. Danilo Gligoroski, Hristina Mihajloska and R. E. Jensen. π-Cipher v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
16. A. B. A. L. F. M.-B. M. N. M. Q. W. Elena Andreeva, Begl Bilgin and K. Yasuda. PRIMATEs v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
17. G. L. C. R.-P. S. Elif Bilge Kavun, Martin M. Lauridsen and T. Yal. PRØST v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
18. P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer, and S. S. Thomsen. Grøstl–a SHA-3 candidate. Submission to NIST, 2008.
19. G. Jakimoski and S. Khajuria. ASC-1: An Authenticated Encryption Stream Cipher. In *Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 356–372. Springer, 2012.
20. M. R. A. Javad Alizadeh and N. Bagheri. Artemia v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
21. P. J. Jean-Philippe Aumasson and S. Neves. NORX v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
22. A. Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In *Advances in Cryptology-CRYPTO 2004*, pages 306–316. Springer, 2004.
23. P. Jovanovic, A. Luykx, and B. Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In *Advances in Cryptology - ASIACRYPT 2014*. Springer.

24. S. Lucks. A failure-friendly design principle for hash functions. In *Advances in Cryptology-ASIACRYPT 2005*, pages 474–494. Springer, 2005.
25. R. S. Manjunath. Provably secure authenticated encryption modes. Masters Thesis, Indraprastha Institute of Information Technology, Delhi, 2013.
26. D. A. McGrew and J. Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.
27. R. C. Merkle. One way hash functions and DES. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 428–446. Springer, 1990.
28. G. V. A. Michal Peeters Guido Bertoni, Joan Daemen and R. V. Keer. KEYAK v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
29. D. Moody, S. Paul, and D. Smith-Tone. Improved Indifferentiability Security Bound for the JH Mode. In *3rd SHA-3 Candidate Conference*, 2012.
30. M. Nandi and S. Paul. Speeding up the wide-pipe: Secure and fast hashing. In *Progress in Cryptology-INDOCRYPT 2010*, pages 144–162. Springer, 2010.
31. NIST. Secure Hash Standard. In Federal Information Processing Standard, FIPS-180, 1993.
32. NIST. Secure Hash Standard. In Federal Information Processing Standard, FIPS-180-1, 1995.
33. S. Paul, E. Homsirikamol, and K. Gaj. A Novel Permutation-based Hash Mode of Operation FP and The Hash Function SAMOSA. In *Progress in Cryptology-INDOCRYPT 2012*, pages 509–527. Springer, 2012.
34. E. H. K. M. J. P.-M. R. M. S. Pawe Morawiecki, Kris Gaj and M. Wjcik. ICEPOLE v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
35. M. P. PGuido Bertoni, Joan Daemen and R. V. K. G. V. Assche. KETJE v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
36. P. Rogaway, M. Bellare, and J. Black. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.
37. M. J. O. Saarinen. CBEAM: Efficient Authenticated Encryption from Feebly One-Way ϕ Functions. In *CT-RSA*, volume 8366 of *Lecture Notes in Computer Science*, pages 251–269. Springer, 2014.
38. M.-J. O. Saarinen. STRIBOB v1. CEASAR Cryptographic Competitions, 2014. `http://competitions.cr.yp.to/caesar-submissions.html`.
39. D. Whiting, N. Ferguson, and R. Housley. Counter with CBC-MAC (CCM). *Request for Comments (RFC)*, (3610), 2003.
40. H. Wu. The Hash Function JH. Submission to NIST (round 3), 2011.
41. H. Wu and B. Preneel. AEGIS: A Fast Authenticated Encryption Algorithm. In *Selected Areas in Cryptography*, volume 8282 of *Lecture Notes in Computer Science*, pages 185–201. Springer, 2013.

Table 5. Game G_0 perfectly simulates $(JHAE - \pi, \pi^{-1})$

Game G_0

Initialize:
$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p - 1$ do:
    $y'_i \parallel y_i = O_2(x'_i \parallel x_i)$;
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p = O_2(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
$v = \pi(m)$
return v
— on O_3-query v—//Inverse Query
$m = \pi^{-1}(v)$
return m

Table 6. In game G_1 the permutations π and π^{-1} are simulated .

Game G_1

Initialize:
$X = \emptyset$;$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p-1$ do:
    $y'_i \parallel y_i = O_2(x'_i \parallel x_i)$;
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p = O_2(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists (m', v') \in X$ S.T $v' = v$ then
$v \leftarrow \{0,1\}^{2n} \backslash \{v' : (m', v') \in X\}$
$X = X \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists (m', v') \in X$ S.T $m' = m$ then
$m \leftarrow \{0,1\}^{2n} \backslash \{m' : (m', v') \in X\}$
$X = X \cup (m, v)$
return m

Table 7. In game G_2 the bad event type-0 may occur.

Game G_2
Initialize:
$X = \emptyset$;$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p-1$ do:
$\quad y'_i \parallel y_i = O_2(x'_i \parallel x_i)$;
$\quad x'_{i+1} = y'_i \oplus m_{i+1}$;
$\quad x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p = O_2(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $v' = v$ then $bad_0 \leftarrow true$
$X = X \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $m' = m$ then $bad_0 \leftarrow true$
$X = X \cup (m, v)$
return m

Table 8. In game G_3 oracle O_2 is simulated inside oracle O_1.

Game G_3

Initialize:
$X = \emptyset$;$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p-1$ do:
    if $(x'_i \parallel x_i, y'_i \parallel y_i) \in X$ then return $y'_i \parallel y_i$
    else $y'_i \parallel y_i \longleftarrow \{0,1\}^{2n}$
    if $\exists((x'_i \parallel x_i)', (y'_i \parallel y_i)') \in X$ S.T $(y'_i \parallel y_i)' = y'_i \parallel y_i$ then $bad_0 \leftarrow true$
    $X = X \cup (x'_i \parallel x_i, y'_i \parallel y_i)$
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
if $(x'_p \parallel x_p, y'_p \parallel y_p) \in X$ then return $y'_p \parallel y_p$
else $y'_p \parallel y_p \longleftarrow \{0,1\}^{2n}$
if $\exists((x'_p \parallel x_p)', (y'_p \parallel y_p)') \in X$ S.T $(y'_p \parallel y_p)' = y'_p \parallel y_p$ then $bad_0 \leftarrow true$
$X = X \cup (x'_p \parallel x_p, y'_p \parallel y_p)$
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $v' = v$ then $bad_0 \leftarrow true$
$X = X \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $m' = m$ then $bad_0 \leftarrow true$
$X = X \cup (m, v)$
return m

Table 9. In game G_4 bad event type-1 may occur.

Game G_4

Initialize:
$X_{O_1} = X_{O_2} = \emptyset$; $X = X_{O_1} \parallel X_{O_2}$; $K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p-1$ do:
    if $(x'_i \parallel x_i, y'_i \parallel y_i) \in X_{O_1}$ then return $y'_i \parallel y_i$
    else if $(x'_i \parallel x_i, y'_i \parallel y_i) \in X_{O_2}$ then $bad_1 \leftarrow true$
    else $y'_i \parallel y_i \longleftarrow \{0,1\}^{2n}$
    if $\exists((x'_i \parallel x_i)', (y'_i \parallel y_i)') \in X$ S.T $(y'_i \parallel y_i)' = y'_i \parallel y_i$ then $bad_0 \leftarrow true$
    $X_{O_1} = X_{O_1} \cup (x'_i \parallel x_i, y'_i \parallel y_i)$
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
if $(x'_p \parallel x_p, y'_p \parallel y_p) \in X_{O_1}$ then return $y'_p \parallel y_p$
else if $(x'_p \parallel x_p, y'_p \parallel y_p) \in X_{O_2}$ then $bad_1 \leftarrow true$
else $y'_p \parallel y_p \longleftarrow \{0,1\}^{2n}$
if $\exists((x'_p \parallel x_p)', (y'_p \parallel y_p)') \in X$ S.T $(y'_p \parallel y_p)' = y'_p \parallel y_p$ then $bad_0 \leftarrow true$
$X_{O_1} = X_{O_1} \cup (x'_p \parallel x_p, y'_p \parallel y_p)$
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $v' = v$ then $bad_0 \leftarrow true$
$X_{O_2} = X_{O_2} \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $m' = m$ then $bad_0 \leftarrow true$
$X_{O_2} = X_{O_2} \cup (m, v)$
return m

Table 10. In G_5 , bad event type-2 may occur.

Game G_5

Initialize:
$X_{O_1} = X_{O_2} = W_{O_1} = W_{O_2} = Y_{O_1} = Y_{O_2} = \emptyset$; $X = X_{O_1} \parallel X_{O_2}$; $W = W_{O_1} \parallel W_{O_2}$; $Y = Y_{O_1} \parallel Y_{O_2}$;
$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p - 1$ do:
    if $(x'_i \parallel x_i, y'_i \parallel y_i) \in X_{O_1}$ then return $y'_i \parallel y_i$
    else if $(x'_i \parallel x_i, y'_i \parallel y_i) \in X_{O_2}$ then $bad_1 \leftarrow true$
    else $y'_i \parallel y_i \longleftarrow \{0,1\}^{2n}$
    if $\exists((x'_i \parallel x_i)', (y'_i \parallel y_i)') \in X$ S.T $(y'_i \parallel y_i)' = y'_i \parallel y_i$ then $bad_0 \leftarrow true$
    $X_{O_1} = X_{O_1} \cup (x'_i \parallel x_i, y'_i \parallel y_i)$
    $W_{O_1} = W_{O_1} \cup (x'_i \parallel x_i)$, $Y_{O_1} = Y_{O_1} \cup (y'_i \parallel y_i)$
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
if $(x'_p \parallel x_p, y'_p \parallel y_p) \in X_{O_1}$ then return $y'_p \parallel y_p$
else if $(x'_p \parallel x_p, y'_p \parallel y_p) \in X_{O_2}$ then $bad_1 \leftarrow true$
else $y'_p \parallel y_p \longleftarrow \{0,1\}^{2n}$
if $\exists((x'_p \parallel x_p)', (y'_p \parallel y_p)') \in X$ S.T $(y'_p \parallel y_p)' = y'_p \parallel y_p$ then $bad_0 \leftarrow true$
$X_{O_1} = X_{O_1} \cup (x'_p \parallel x_p, y'_p \parallel y_p)$
$W_{O_1} = W_{O_1} \cup (x'_p \parallel x_p)$, $Y_{O_1} = Y_{O_1} \cup (y'_p \parallel y_p)$
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X_{O_2}$ then return v
if $m \in W_{O_1}$ then $bad_2 \leftarrow true$
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $v' = v$ then $bad_1 \leftarrow true$
$X_{O_2} = X_{O_2} \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X_{O_2}$ then return m
if $v \in Y_{O_1}$ then $bad_2 \leftarrow true$
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists(m', v') \in X$ S.T $m' = m$ then $bad_1 \leftarrow true$
$X_{O_2} = X_{O_2} \cup (m, v)$
return m

Table 11. In game G_6 O_1 does not keeps the history of intermediate queries.

Game G_6

Initialize:
$X = \emptyset$;$K \longleftarrow \{0,1\}^n$;
$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
— on O_1 -query (N,A,M) —
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
for $i = 0$ to $p-1$ do:
    $y'_i \parallel y_i \leftarrow \{0,1\}^{2n}$;
    $x'_{i+1} = y'_i \oplus m_{i+1}$;
    $x_{i+1} = y_i \oplus m_i$
end for
$y'_p \parallel y_p \leftarrow \{0,1\}^{2n}$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
$T = x_{p+1} \oplus K$
Return (C,T)
— on O_2-query m—
if $(m,v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
$X = X \cup (m,v)$
return v
— on O_3-query v—//Inverse Query
if $(m,v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
$X = X \cup (m,v)$
return m

Table 12. G_7 (boxes removed) and G_8 (boxes included). In game G_7 , blocks of ciphertext and tag value are generated randomly. In game G_8 there is a switch from random permutation to random function.

Games G_7 and $\boxed{G_8}$
Initialize: $X = \emptyset$ — on O_1-query (N,A,M)— $pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$ for $i = 1$ to p do: $x'_i \longleftarrow \{0,1\}^n$ end for $T \longleftarrow \{0,1\}^n$ $C = x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$ Return (C, T) — on O_2-query m— if $(m, v) \in X$ then return v else $v \longleftarrow \{0,1\}^{2n}$ $\boxed{\text{if } \exists (m', v') \in X \text{ S.T } v' = v \text{ then}}$ $\boxed{v \leftarrow \{0,1\}^{2n} \backslash \{v' : (m', v') \in X\}}$ $X = X \cup (m, v)$ return v — on O_3-query v—//Inverse Query if $(m, v) \in X$ then return m else $m \longleftarrow \{0,1\}^{2n}$ $\boxed{\text{if } \exists (m', v') \in X \text{ S.T } m' = m \text{ then}}$ $\boxed{m \leftarrow \{0,1\}^{2n} \backslash \{m' : (m', v') \in X\}}$ $X = X \cup (m, v)$ return m

Table 13. Game G_9 perfectly simulates an ideal AE, *i.e*, RO, π and π^{-1}.

Game G_9
Initialize:
$X = \emptyset$
— on O_1-query (N,A,M)—
$pad(A) \| pad(M) = m_1 \parallel m_2 \parallel ... \parallel m_p$
$C \longleftarrow \{0,1\}^{\lvert Pad(M)\rvert}$
$T \longleftarrow \{0,1\}^n$
Return (C, T)
— on O_2-query m—
if $(m, v) \in X$ then return v
else $v \longleftarrow \{0,1\}^{2n}$
if $\exists (m', v') \in X$ S.T $v' = v$ then
$v \leftarrow \{0,1\}^{2n} \backslash \{v' : (m', v') \in X\}$
$X = X \cup (m, v)$
return v
— on O_3-query v—//Inverse Query
if $(m, v) \in X$ then return m
else $m \longleftarrow \{0,1\}^{2n}$
if $\exists (m', v') \in X$ S.T $m' = m$ then
$m \leftarrow \{0,1\}^{2n} \backslash \{m' : (m', v') \in X\}$
$X = X \cup (m, v)$
return m