Journal of Computing and Security

January 2015, Volume 2, Number 1 (pp. 3–20)

http://www.jcomsec.org

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

Javad Alizadeh [a,*], Mohammad Reza Aref [b], Nasour Bagheri [c,d], Alireza Rahimi [a]

[a] *Faculty of Communication and Information Technology, Imam Hossein University, Tehran, Iran.*

[b] *Information Systems and Security Lab. (ISSL), Electrical Eng. Department, Sharif University of Technology, Tehran, Iran.*

[c] *Electrical Engineering Department of Shahid Rajaee Teacher Training University, Tehran, Iran.*

[d] *School of Computer Science of Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.*

ARTICLE INFO.

Article history:
Received: 12 April 2014
Revised: 15 December 2015
Accepted: 9 January 2016
Published Online: 7 February 2016

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

ABSTRACT

Authenticated encryption (*AE*) schemes provide both privacy and integrity of data. CAESAR is a competition to design and analysis of the *AE* schemes. An *AE* scheme has two components: a mode of operation and a primitive. In this paper JHAE, a novel authenticated encryption mode, is presented based on the JH (SHA-3 finalist) hash mode. JHAE is an on-line and single-pass dedicated *AE* mode based on permutation that supports optional associated data (AD). It is proved that this mode, based on ideal permutation, achieves privacy and integrity up to $O(2^{n/2})$ queries where the length of the used permutation is $2n$. To decrypt, JHAE does not require the inverse of its underlying permutation and therefore saves area space. JHAE has been used by Artemia, one of the CAESAR's first round candidates.

1 Introduction

An authenticated encryption scheme (*AE*) can establish privacy and authentication, simultaneously. The schemes are important since in many applications, such as Transport Layer Security (TLS), the two main goals in information security must be established simultaneously [1]. Now, the NIST-funded CAESAR competition for *AE* [2] which has been held by International Association for Cryptologic Research (IACR), has attracted more attention to the *AE*.

One approach (the first approach) to designing an *AE* scheme is the combining two algorithms which one of them provides confidentiality and the other provides authenticity. The schemes were named generic compositions [1]. In the approach two separate algorithms with two different keys are required. Then this approach is not efficient. To improve the efficiency of the *AE* schemes based on a generic composition, the *AE* schemes based on a block cipher mode were proposed. In the schemes a block cipher is used in a special mode [3–5]. Although the schemes solved the problem of requiring two separate algorithms in the generic composition schemes, but the necessity for a running the full round block cipher to process each message block in the modes reduce the efficiency of the schemes. To solve this problem and enhance the efficiency of the *AE* schemes based on a block cipher mode, dedicated *AE* schemes were proposed [6–11].

A dedicated *AE* scheme has two main components: an special mode of operation and a primitive such as a random permutation or random function which is used in the mode. Therefore, in designing a new dedicated *AE* one can consider two main stages [12]: designing a new dedicated mode and designing a new

* Corresponding author.

Email addresses: jaalizadeh@ihu.ac.ir (J. Alizadeh), aref@sharif.edu (M. R. Aref), nbagheri@srttu.edu (N. Bagheri), arahimi@ihu.ac.ir (A. R. Rahimi)

ISSN: 2322-4460

random permutation or random function to be used in the mode.

Extending a hash function mode to a dedicated *AE* mode is a general approach to design a new *AE* mode. For example, duplex constructions [17], which were used in designing of the CAESAR candidates Ascon [18], ICEPOLE [19], KETJE [20], KEYAK [21], NORX [22], π-Cipher [23], PRIMATEs-GIBBON [24], PRIMATEs-HANUMAN [24], PRIMATEs-APE [24], PRØST-APE [25], and STRIBOB [26], are closely related to the sponge construction [27]. Other examples include FWPAE and FPAE modes [28] that were obtained from FWP [29] and FP [13] hash function modes, respectively. Also, PPAE [30] is a new *AE* mode based on Parazoa hash [31] construction. An important challenge in developing an *AE* mode from another mode (e.g. hash mode) is to prove its security, to ensure that transition the hash mode to another application does not make any structural flaws. Although obtaining an *AE* mode from a hash mode is not complex task, but providing security bounds for the new mode is not straighted forward.

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 [32, 33] was used in designing of many famous hash functions such as SHA-0 [34] and SHA-1 [35]. Some flaws in the construction (e.g. multi-collision attack [36]) leads to development of new hash constructions such as Wide-pipe [37], Sponge [27], JH [38], Grøstl [15], FP [13], and Parazoa [31]. The last five 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 [39] which was the winner of the competition. JH mode is similar to the Sponge mode with these differences that in the JH mode, the length of used permutation is the twice of the length of message blocks and the message blocks are added to the rate and capacity sections of the mode. So, the efficiency of JH mode in comparison with the Sponge one, is low.

A comparison of some hash function modes was presented in [13]. Also, a compression of SHA-3 finalists hash modes was presented in [40]. For the modes Sponge, Grøstl, JH, and FP the comparison was summarized from [13] in Table 1 where ϵ is a small fraction due to the preimage attack on JH presented in [41]. 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 [13].

Contribution

In this paper JH hash function mode [38] is used to develop a new dedicated *AE* mode, called JHAE. The motivation for designing JHAE, is the CAESAR competition and the main reasons of using JH mode to design a new *AE* mode are given as follows:

- It is a permutation-based mode.
- Keccak (which uses the Sponge construction), Grøstl, and JH are three finalists of the SHA-3 competition. Compared by Grøstl, JH uses only one permutation and compared by Sponge, it has better indifferentiability upper bound (See Table 1).
- Duplex constructions [17], FPAE [28], and recently PPAE [30] are three *AE* modes based on the Sponge, FP, and Parazoa hash function modes, respectively, and so far no *AE* mode has been presented based on the JH hash function mode.
- Extensive researches on the JH hash mode had done during SHA-3 competition and they have shown that there was no significant vulnerability in this hash 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 is proved in this paper that the mode achieves 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 is $2n$. In addition, it is demonstrated that the integrity bound of JHAE is reduced to the indifferentiability of JH hash mode, which is at least $O(2^{n/2})$.

JHAE in the CAESAR Competition

Artemia [12, 42] is a family of the dedicated authenticated encryption schemes which was submitted to the CAESAR competition. It is a sponge-based authenticated encryption scheme that uses the JHAE mode. Exclude Artemia, all of the sponge-based candidates of CAESAR use the duplex constructions [43]. Until now (in the duration of the CAESAR competition) no flaw has been reported for JHAE and Artemia. Some of the works in the duration of CAESAR which were cited JHAE and Artemia are as follows:

- In [44], Jovanovic et. al. showed that sponge based constructions for authenticated encryption can achieve a significantly higher bound than $2^{c/2}$, where c is their capacity. (Note that the capacity of JHAE, is n). They proved that NORX [22], a CAESAR candidate, achieves this bound. They also showed how to apply their proof

Table 1. Comparison of some permutation-based hash modes [13].

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	[14]
Grøstl	n	$2n$	0.5	$n/2$	n	2	[15]
JH	n	$2n$	0.5	$n/2$	$n(1-\epsilon)$	1	[16]
FP	n	$2n$	0.5	$n/2$	n	1	[13]

to seven other Sponge-based CAESAR submissions: Ascon, CBEAM/STRIBOB, ICEPOLE, Keyak, PRIMATEs-GIBBON, and PRIMATEs-HANUMAN. It was mentioned in [44] that the security proofs may be applicable for the modes of Artemia (e.g. JHAE) and π-Cipher. JHAE is slightly different from the seven modes, therefore, a generalization of the proof of [44] to JHAE is not entirely straightforward.

- In [45] Agrawal et. al. proposed a new sponge-based *AE* technique for handling long ciphertexts in memory constrained devices. They considered all of the nine submissions to the CAESAR which have the sponge construction in their generalized strate. The results of [45] shows that only two schemes Ascon and PRIMATEs-GIBBON of the nine sponge-based schemes satisfy the constraints in [45] and suitable for limited memory applications.
- In [46], Hoang et. al. analysed the submissions of the CAESAR by assuming that the nonce (in the schemes) can be repeated. With respect to this assumption, they presented some attacks on the submissions (e.g. Artemia). Since Artemia is a nonce respecting scheme then the attack in [46] does not affect the security of Artemia.
- In [47], Andreeva et al. studied the security of the keyed sponge-based constructions such as JHAE and presented the improved indefferntiablity bound for some of the constructions. Their results shows that the indefferntiablity bound of JHAE can be improved.

The performance of JHAE and other sponge-based *AE* modes which were submitted to the CAESAR can be compared with respect to [45]. A comparison between Artemia and other dedicated *AE* schemes which were submitted to the CAESAR competition was presented in [43]. In addition to, the comparison between performance of Artemia and other CAESAR submissions can be found in [48]. With respect to [43], 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).

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 Figure 1, is described. JHAE is developed from JH hash function mode (Figure 2) [38] and iterates a fixed permutation $\pi : \{0,1\}^{2n} \to \{0,1\}^{2n}$. 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 Algorithm 1. 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 3):

(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

Table 2. Comparison between Artemia and other sponge-based candidates of CAESAR [43]. n.n. means unnamed custom primitive.

Sponge-Based *AE*	Design	Primitive	Security Proofs	Parallelizable	On-Line	Nonce Misuse Resistance	Inverse-Free	Reference
Artemia	JHAE	Artemia	✓	×	✓	×	✓	[42]
Ascon	Duplex	Ascon	✓	×	✓	✓	✓	[18]
ICEPOLE	Duplex	Keccak-like	✓	✓	✓	✓	✓	[19]
KETJE	Duplex	Keccak-f	✓	×	✓	×	✓	[20]
KEYAK	Duplex	Keccak-f	✓	✓	✓	×	✓	[21]
NORX	Duplex	n.n.	✓	✓	✓	×	✓	[22]
π-Cipher	Duplex	n.n.	×	✓	✓	×	✓	[23]
PRIMATEs-GIBBON	Duplex	PRIMATE	✓	×	✓	×	✓	[24]
PRIMATEs-HANUMAN	Duplex	PRIMATE	✓	×	✓	×	✓	[24]
PRIMATEs-APE	Duplex	PRIMATE	✓	×	✓	✓	×	[24]
PRØST-APE	Duplex	PRØST	×	×	✓	✓	×	[25]
STRIBOB	Duplex	Streebog	✓	×	✓	×	✓	[26]

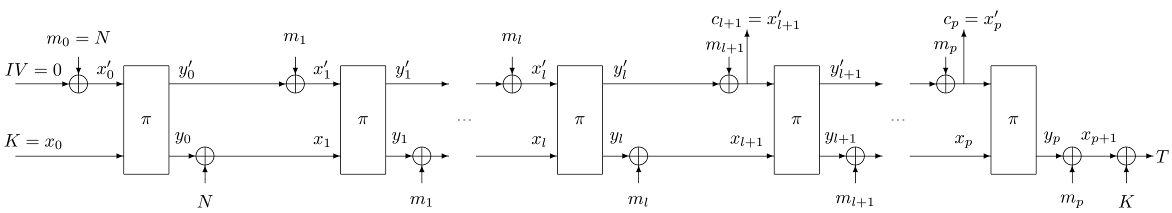

Figure 1. JHAE mode of operation (encryption and authentication), where $pad(A) = m_1 \| m_2 \| \ldots \| m_l$ and $pad(M) = m_{l+1} \parallel m_{l+2} \parallel ... \parallel m_p$

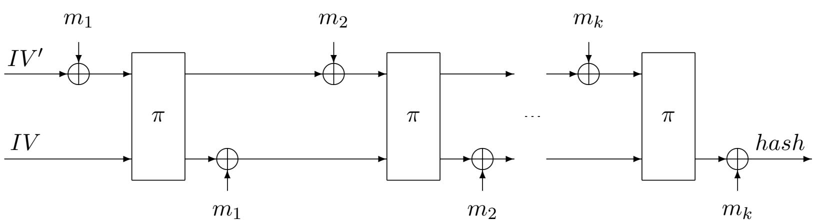

Figure 2. JH hash mode [16]

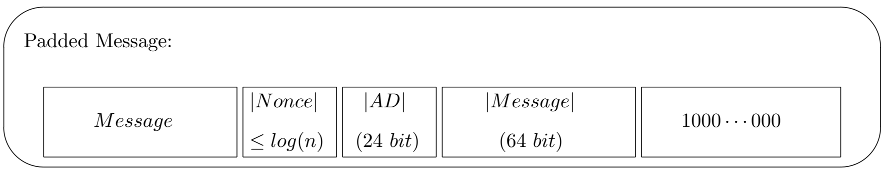

Figure 3. Message padding in JHAE

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 4). 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.

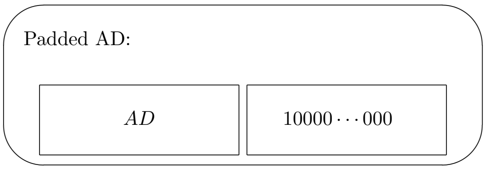

Figure 4. AD padding in JHAE

Algorithm 1 Encryption and authentication using JHAE

procedure $JHAE - E^{\pi}(K, N, M, A)$
  $m_1 \| m_2 \| \ldots \| m_l \leftarrow pad(A)$
  $m_{l+1} \parallel m_{l+2} \parallel \ldots \parallel m_p \leftarrow pad(M)$
  $IV \leftarrow 0$
  $m_0 \leftarrow N$
  $x'_0 \leftarrow IV \oplus m_0$
  $x_0 \leftarrow K$
  for $i \leftarrow 0, p-1$ **do**
    $y'_i \parallel y_i \leftarrow \pi(x'_i \parallel x_i)$
    $x'_{i+1} \leftarrow y'_i \oplus m_{i+1}$
    $x_{i+1} \leftarrow y_i \oplus m_i$
  end for
  $y'_p \parallel y_p \leftarrow \pi(x'_p \parallel x_p)$
  $x_{p+1} \leftarrow y_p \oplus m_p$
  $C \leftarrow x'_{l+1} \parallel x'_{l+2} \parallel \ldots \parallel x'_p$
  $T \leftarrow x_{p+1} \oplus K$
  return (C, T) ▷ C is the ciphertext and T is the authentication tag
end procedure

2.2 Decryption and Verification

JHAE decryption-verification procedure, depicted in Algorithm 2, 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 [49] 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 $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$, is (t_A, σ, ϵ)-indistinguishable*

Algorithm 2 Decryption and verification using JHAE

procedure $JHAE - D^{\pi}(K, N, C, T, A)$
  $m_1 \| m_2 \| \ldots \| m_l \leftarrow pad(A)$
  $c_1 \parallel c_2 \parallel \ldots \parallel c_p \leftarrow C$
  $IV \leftarrow 0$
  $m_0 \leftarrow N$
  $x'_0 \leftarrow IV \oplus m_0$
  $x_0 \leftarrow K$
  $x'_{l+1} \parallel x'_{l+2} \parallel \ldots \parallel x'_{l+p} \leftarrow c_1 \parallel c_2 \parallel \ldots \parallel c_p$
  for $i \leftarrow 0, l-1$ **do**
    $y'_i \parallel y_i \leftarrow \pi(x'_i \parallel x_i)$
    $x'_{i+1} \leftarrow y'_i \oplus m_{i+1}$
    $x_{i+1} \leftarrow y_i \oplus m_i$
  end for
  for $i \leftarrow l, p-1$ **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
  $y'_p \parallel y_p \leftarrow \pi(x'_p \parallel x_p)$
  $x_{p+1} \leftarrow y_p \oplus m_p$
  $M \leftarrow m_{l+1} \parallel m_{l+2} \parallel \ldots \parallel m_p$
  $T' \leftarrow x_{p+1} \oplus K$
  if $T' = T$ **then**
    return M ▷ M is the plaintext
  else
    return $\perp$
  end if
end procedure

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, $\epsilon \leq \frac{\sigma(\sigma-1)}{2^{2n-1}} + \frac{\sigma^2}{2^{2n}} + \frac{\sigma^2}{2^n}$, 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 $\mathcal{A}$.

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 Algorithm 3). 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 Algorithm 4). 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 [49] is done in G_1(see Algorithm 5). 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 Algorithm 6). 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 Algorithm 7) 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 Algorithm 8), 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 Algorithm 9) 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 Algorithm 10), 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 Algorithm 11), a PRF-PRP switch [49] 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 Algorithm 12) 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 [49], the following can be stated:

$$\begin{aligned}
&Adv_{JHAE}^{Privacy}(\mathcal{A}) \\
&= Pr[\mathcal{A}^{JHAE-E,\pi,\pi^{-1}} \Rightarrow 1] - Pr[\mathcal{A}^{RO,\pi,\pi^{-1}} \Rightarrow 1] \\
&= Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_9} \Rightarrow 1] \\
&= (Pr[\mathcal{A}^{G_0} \Rightarrow 1] - Pr[\mathcal{A}^{G_1} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_1} \Rightarrow 1] - Pr[\mathcal{A}^{G_2} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_2} \Rightarrow 1] - Pr[\mathcal{A}^{G_3} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_3} \Rightarrow 1] - Pr[\mathcal{A}^{G_4} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_4} \Rightarrow 1] - Pr[\mathcal{A}^{G_5} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_5} \Rightarrow 1] - Pr[\mathcal{A}^{G_6} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_6} \Rightarrow 1] - Pr[\mathcal{A}^{G_7} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_7} \Rightarrow 1] - Pr[\mathcal{A}^{G_8} \Rightarrow 1]) \\
&+(Pr[\mathcal{A}^{G_8} \Rightarrow 1] - Pr[\mathcal{A}^{G_9} \Rightarrow 1]) \\
&\leq 0 + \frac{\sigma(\sigma-1)}{2^{2n+1}} + 0 + \frac{\sigma^2}{2^{2n}} + \frac{\sigma^2}{2^n} + \frac{\sigma(\sigma-1)}{2^{2n}} + 0 \\
&+ \frac{\sigma(\sigma-1)}{2^{2n+1}} + 0 \leq \frac{\sigma(\sigma-1)}{2^{2n-1}} + \frac{\sigma^2}{2^{2n}} + \frac{\sigma^2}{2^n}.
\end{aligned}$$

□

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 $S = (N_i, (A\|C)_i, T_i)$ where $1 \leq i \leq q$ 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:

$$Pr[\mathcal{A}^{INT}_{JHAE} \Rightarrow 1] = Pr[Case\ 001] + Pr[Case\ 010] + Pr[Case\ 011] + Pr[Case\ 100] + Pr[Case\ 101] + Pr[Case\ 110] + Pr[Case\ 111]. \quad (1)$$

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 Figure 2 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 $\frac{\sigma^2}{2^n}$ [16], 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):

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

□

Comparing the security of JHAE and JH

In [41], Bhattacharyya et al. showed that in the ideal permutation model, JH is indifferentiable from a random oracle. They used the approach of Chang and Nandi in [50]. Andreeva et al. in [40] showed that the bounds for JH is not accurate when the security of preimage and second preimage are considered. For this, they considered the JH feathers an used a direct approach. Finally, Moody et al. in [16] improved the indifferentiability bound for JH. They used three games in the game playing framework. The results of [16] were summarized in Table 1.

In this paper, the game playing framework was used to find an indefferntiability bound for JHAE. The bound is $2^{n/2}$ and similar to the bound of JH in [16]. This is the first nontrivial security bound for JHAE and can be improved using the technique in [44].

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 1, 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 $q = O(2^{n/2})$. 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.

For a future work, the security bound of JHAE can be improved using the security model introduced in [44].

Acknowledgment

This work was partially supported by Iran-NSF under grant no. 92.32575.

Appendix A Sequence of Games

Algorithm 3 Game G_0 perfectly simulates $(JHAE-\pi, \pi^{-1})$

procedure INITIALIZATION
    $K \leftarrow \{0,1\}^n$
    $IV \leftarrow 0$
    $m_0 \leftarrow N$
    $x'_0 \leftarrow IV \oplus m_0$
    $x_0 \leftarrow K$
end procedure
procedure O_1 -QUERY(N, A, M)
    $m_1 \parallel m_2 \parallel ... \parallel m_p \leftarrow pad(A) \| pad(M)$
    for $i \leftarrow 0, p-1$ **do**
        $y'_i \parallel y_i \leftarrow O_2(x'_i \parallel x_i)$
        $x'_{i+1} \leftarrow y'_i \oplus m_{i+1}$
        $x_{i+1} \leftarrow y_i \oplus m_i$
    end for
    $y'_p \parallel y_p \leftarrow O_2(x'_p \parallel x_p)$
    $x_{p+1} \leftarrow y_p \oplus m_p$
    $C \leftarrow x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
    $T \leftarrow x_{p+1} \oplus K$
    return (C, T)
end procedure
procedure O_2-QUERY(m)
    $v \leftarrow \pi(m)$
    return v
end procedure
procedure O_3-QUERY(v)
    $m \leftarrow \pi^{-1}(v)$
    return m
end procedure

Algorithm 4 In game G_1 the permutations π and π^{-1} are simulated .

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

Algorithm 5 In game G_2 the bad event type-0 may occur.

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

Algorithm 6 In game G_3 oracle O_2 is simulated inside oracle O_1.

```
procedure INITIALIZATION
    X = ∅
    K ← {0,1}^n
    IV ← 0
    m_0 ← N
    x'_0 ← IV ⊕ m_0
    x_0 ← K
end procedure
procedure O_1 -QUERY(N, A, M)
    m_1 || m_2 || ... || m_p ← pad(A)||pad(M)
    for i ← 0, p − 1 do
        if (x'_i || x_i, y'_i || y_i) ∈ X then
            return y'_i || y_i
        else
            y'_i || y_i ← {0,1}^{2n}
        end if
        if ∃((x'_i || x_i)', (y'_i || y_i)') ∈ X S.T (y'_i || y_i)' = y'_i || y_i then
            bad_0 ← true
        end if
        X ← X ∪ (x'_i || x_i, y'_i || y_i)
        x'_{i+1} ← y'_i ⊕ m_{i+1}
        x_{i+1} ← y_i ⊕ m_i
    end for
    if (x'_p || x_p, y'_p || y_p) ∈ X then
        return y'_p || y_p
    else
        y'_p || y_p ← {0,1}^{2n}
    end if
    if ∃((x'_p || x_p)', (y'_p || y_p)') ∈ X S.T (y'_p || y_p)' = y'_p || y_p then
        bad_0 ← true
    end if
    X ← X ∪ (x'_p || x_p, y'_p || y_p)
    x_{p+1} ← y_p ⊕ m_p
    C ← x'_{l+1} || x'_{l+2} || ... || x'_p
    T ← x_{p+1} ⊕ K
    return (C, T)
end procedure
procedure O_2-QUERY(m)
    if (m, v) ∈ X then
        return v
    else
        v ← {0,1}^{2n}
    end if
    if ∃(m', v') ∈ X S.T v' = v then
        bad_0 ← true
        X = X ∪ (m, v)
    end if
    return v
end procedure
procedure O_3-QUERY(v)
    if (m, v) ∈ X then
        return m
    else
        m ← {0,1}^{2n}
    end if
    if ∃(m', v') ∈ X S.T m' = m then
        bad_0 ← true
        X = X ∪ (m, v)
    end if
    return m
end procedure
```

Algorithm 7 In game G_4 bad event type-1 may occur.

procedure INITIALIZATION
    $X_{O_1} \leftarrow \emptyset$
    $X_{O_2} \leftarrow \emptyset$
    $X \leftarrow X_{O_1} \parallel X_{O_2}$
    $K \leftarrow \{0,1\}^n$
    $IV \leftarrow 0$
    $m_0 \leftarrow N$
    $x'_0 \leftarrow IV \oplus m_0$
    $x_0 \leftarrow K$
end procedure
procedure O_1 -QUERY(N, A, M)
    $m_1 \parallel m_2 \parallel ... \parallel m_p \leftarrow pad(A) \| pad(M)$
    for $i \leftarrow 0, p-1$ **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
            $y'_i \parallel y_i \leftarrow \{0,1\}^{2n}$
        end if
        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**
            $bad_0 \leftarrow true$
        end if
        $X_{O_1} \leftarrow X_{O_1} \cup (x'_i \parallel x_i, y'_i \parallel y_i)$
        $x'_{i+1} \leftarrow y'_i \oplus m_{i+1}$
        $x_{i+1} \leftarrow y_i \oplus m_i$
    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
        $y'_p \parallel y_p \leftarrow \{0,1\}^{2n}$
    end if
    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**
        $bad_0 \leftarrow true$
    end if
    $X_{O_1} \leftarrow X_{O_1} \cup (x'_p \parallel x_p, y'_p \parallel y_p)$
    $x_{p+1} \leftarrow y_p \oplus m_p$
    $C \leftarrow x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
    $T \leftarrow x_{p+1} \oplus K$
    return (C, T)
end procedure
procedure O_2-QUERY(m)
    if $(m, v) \in X$ **then**
        return v
    else
        $v \leftarrow \{0,1\}^{2n}$
    end if
    if $\exists(m', v') \in X$ S.T $v' = v$ **then**
        $bad_0 \leftarrow true$
        $X = X \cup (m, v)$
    end if
    return v
end procedure
procedure O_3-QUERY(v)
    if $(m, v) \in X$ **then**
        return m
    else
        $m \leftarrow \{0,1\}^{2n}$
    end if
    if $\exists(m', v') \in X$ S.T $m' = m$ **then**
        $bad_0 \leftarrow true$
        $X = X \cup (m, v)$
    end if
    return m
end procedure

Algorithm 8 In G_5 , bad event type-2 may occur.

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

Algorithm 9 In game G_6 O_1 does not keeps the history of intermediate queries.

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

Algorithm 10 In game G_7 , blocks of ciphertext and tag value are generated randomly.

procedure INITIALIZATION
    $X \leftarrow \emptyset$
end procedure
procedure O_1 -QUERY(N, A, M)
    $m_1 \parallel m_2 \parallel ... \parallel m_p \leftarrow pad(A) \| pad(M)$
    for $i \leftarrow 0, p-1$ **do**
        $x'_i \leftarrow \{0,1\}^n$
    end for
    $C \leftarrow x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
    $T \leftarrow \{0,1\}^n$
    return (C, T)
end procedure
procedure O_2-QUERY(m)
    if $(m, v) \in X$ **then**
        return v
    else
        $v \leftarrow \{0,1\}^{2n}$
    end if
    $X = X \cup (m, v)$
    return v
end procedure
procedure O_3-QUERY(v)
    if $(m, v) \in X$ **then**
        return m
    else
        $m \leftarrow \{0,1\}^{2n}$
    end if
    $X = X \cup (m, v)$
    return m
end procedure

Algorithm 11 In game G_8 there is a switch from random function to random permutation .

procedure INITIALIZATION
  $X \leftarrow \emptyset$
end procedure
procedure O_1 -QUERY(N, A, M)
  $m_1 \parallel m_2 \parallel ... \parallel m_p \leftarrow pad(A) \| pad(M)$
  for $i \leftarrow 0, p-1$ **do**
    $x'_i \leftarrow \{0,1\}^n$
  end for
  $C \leftarrow x'_{l+1} \parallel x'_{l+2} \parallel ... \parallel x'_p$
  $T \leftarrow \{0,1\}^n$
  return (C, T)
end procedure
procedure O_2-QUERY(m)
  if $(m, v) \in X$ **then**
    return v
  else
    $v \leftarrow \{0,1\}^{2n}$
  end if
  if $\exists (m', v') \in X$ S.T $v' = v$ **then**
    $v \leftarrow \{0,1\}^{2n} \backslash \{v' : (m', v') \in X\}$
  end if
  $X = X \cup (m, v)$
  return v
end procedure
procedure O_3-QUERY(v)
  if $(m, v) \in X$ **then**
    return m
  else
    $m \leftarrow \{0,1\}^{2n}$
  end if
  if $\exists (m', v') \in X$ S.T $m' = m$ **then**
    $m \leftarrow \{0,1\}^{2n} \backslash \{m' : (m', v') \in X\}$
  end if
  $X = X \cup (m, v)$
  return m
end procedure

Algorithm 12 Game G_9 perfectly simulates an ideal AE, *i.e*, RO, π and π^{-1}.

procedure INITIALIZATION
  $X \leftarrow \emptyset$
end procedure
procedure O_1 -QUERY(N, A, M)
  $m_1 \parallel m_2 \parallel ... \parallel m_p \leftarrow pad(A) \| pad(M)$
  $C \leftarrow \{0,1\}^{|Pad(M)|}$
  $T \leftarrow \{0,1\}^n$
  return (C, T)
end procedure
procedure O_2-QUERY(m)
  if $(m, v) \in X$ **then**
    return v
  else
    $v \leftarrow \{0,1\}^{2n}$
  end if
  if $\exists (m', v') \in X$ S.T $v' = v$ **then**
    $v \leftarrow \{0,1\}^{2n} \backslash \{v' : (m', v') \in X\}$
  end if
  $X = X \cup (m, v)$
  return v
end procedure
procedure O_3-QUERY(v)
  if $(m, v) \in X$ **then**
    return m
  else
    $m \leftarrow \{0,1\}^{2n}$
  end if
  if $\exists (m', v') \in X$ S.T $m' = m$ **then**
    $m \leftarrow \{0,1\}^{2n} \backslash \{m' : (m', v') \in X\}$
  end if
  $X = X \cup (m, v)$
  return m
end procedure

References

[1] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptology*, 21(4):469–491, 2008.

[2] CAESAR. CAESAR: Competition for Authenticated Encryption: Security, Applicability,and Robustness, 2013. http://competitions.cr.yp.to/caesar.html.

[3] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[4] David A. McGrew and John 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.

[5] Doug Whiting, Niels Ferguson, and Russell Housley. Counter with CBC-MAC (CCM). *Request for Comments (RFC)*, (3610), 2003.

[6] Goce Jakimoski and Samant 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.

[7] Andrey Bogdanov, Florian Mendel, Francesco Regazzoni, Vincent Rijmen, and Elmar Tischhauser. ALE: AES-based lightweight authenticated encryption. Preproceedings of Fast Software Encryption (FSE 2013), 2013. To Appear.

[8] Hongjun Wu and Bart 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.

[9] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju 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.

[10] Markku Juhani 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.

[11] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. Preproceedings of Fast Software Encryption (FSE 2014), 2014. To Appear.

[12] Javad Alizadeh, Mohammad Reza Aref, and Nasour Bagheri. Artemia: A Family of Provably Secure Authenticated Encryption Schemes. *The ISC Int'l Journal of Information Security (ISecure)*, 6(2):125–139, 2014.

[13] Souradyuti Paul, Ekawat Homsirikamol, and Kris 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.

[14] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In *Advances in Cryptology–EUROCRYPT 2008*, pages 181–197. Springer, 2008.

[15] Praveen Gauravaram, Lars R Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S Thomsen. Grøstl–a SHA-3 candidate. Submission to NIST, 2008.

[16] Dustin Moody, Souradyuti Paul, and Daniel Smith-Tone. Improved Indifferentiability Security Bound for the JH Mode. In *3rd SHA-3 Candidate Conference*, 2012.

[17] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van 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.

[18] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schlffer. Ascon v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[19] Pawe Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wjcik. ICEPOLE v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[20] PGuido Bertoni, Joan Daemen, Michal Peeters, and Ronny Van Keer Gilles Van Assche. KETJE v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[21] Michal Peeters Guido Bertoni, Joan Daemen, Gilles Van Assche, , and Ronny Van Keer. KEYAK v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[22] Jean Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[23] Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Hkon Jacobsen, Mohamed

El-Hadedy, and Rune Erlend Jensen. π-Cipher v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[24] Elena Andreeva, Begl Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[25] Elif Bilge Kavun, Martin M. Lauridsen, Gregor Leander, Christian Rechberger, Peter Schwabe, and Tolga Yal. PRØST v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[26] Markku Juhani O. Saarinen. STRIBOB v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[27] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge Functions. ECRYPT hash workshop, 2007.

[28] R Sumesh Manjunath. Provably secure authenticated encryption modes. Masters Thesis, Indraprastha Institute of Information Technology, Delhi, 2013.

[29] Mridul Nandi and Souradyuti Paul. Speeding up the wide-pipe: Secure and fast hashing. In *Progress in Cryptology-INDOCRYPT 2010*, pages 144–162. Springer, 2010.

[30] Donghoon Chang, Sumesh Manjunath R, and Somitra Kumar Sanadhya. PPAE: Practical Parazoa Authenticated Encryption Family. In *ProvSec*, pages 198–211. Springer, 2015.

[31] Andreeva Elena, Bart Mennink, and Bart Preneel. The parazoa family: generalizing the sponge hash functions. *International Journal of Information Security*, 11(3):149–165, 2012.

[32] Ralph C Merkle. One way hash functions and DES. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 428–446. Springer, 1990.

[33] Ivan Bjerre Damgård. A design principle for hash functions. In *Advances in CryptologyCRYPTO89 Proceedings*, pages 416–427. Springer, 1990.

[34] NIST. Secure Hash Standard. In Federal Information Processing Standard, FIPS-180, 1993.

[35] NIST. Secure Hash Standard. In Federal Information Processing Standard, FIPS-180-1, 1995.

[36] Antoine Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In *Advances in Cryptology-CRYPTO 2004*, pages 306–316. Springer, 2004.

[37] Stefan Lucks. A failure-friendly design principle for hash functions. In *Advances in Cryptology-ASIACRYPT 2005*, pages 474–494. Springer, 2005.

[38] Hongjun Wu. The Hash Function JH. Submission to NIST (round 3), 2011.

[39] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. Submission to NIST, 2009.

[40] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Skrobot. Security analysis and comparison of the SHA-3 finalists BLAKE, Grstl, JH, Keccak, and Skein. In *Progress in Cryptology-AFRICACRYPT 2012*, pages 287–305. Springer, 2012.

[41] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security analysis of the mode of JH hash function. In *Fast Software Encryption*, pages 168–191. Springer, 2010.

[42] Javad Alizadeh, Mohammad Reza Aref, and Nasour Bagheri. Artemia v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/caesar-submissions.html.

[43] Farzaneh Abed, Christian Forler, and Stefan Lucks. General Overview of the First-Round CAESAR Candidates for Authenticated Ecryption. IACR Cryptology ePrint Archive, 2014. URL http://eprint.iacr.org/2014/792.

[44] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In *Advances in Cryptology - ASIACRYPT 2014*. Springer, 2014.

[45] Megha Agrawal, Donghoon Chang, and Somitra Sanadhya. sp-AELM: Sponge based Authenticated Encryption Scheme for Memory Constrained Devices. In *ACISP 2015*, pages 451–468. Springer, 2015.

[46] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In *CRYPTO 2015*, 2015.

[47] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of keyed sponge constructions using a modular proof approach. In *FSE*, 2015.

[48] CAESAR Candidates Speed Comparison, 2014. http://www1.spms.ntu.edu.sg/~syllab/speed/.

[49] Mihir Bellare and Phillip 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.

[50] Donghoon Chang and Mridul Nandi. Improved indifferentiability security analysis of chopMD hash function. In *FSE*, 2008.

Javad Alizadeh received M.S. degree in Telecommunication in the field of Cryptography from Imam Hossein University, Tehran, Iran, in 2010. He serves as a member of Information Systems and Security Lab (ISSL) at the Electrical Engineering Department of Sharif University of Technology. He is currently working toward the Ph.D. degree in Cryptography at Imam Hossein University. His research interest include Symmetric Cryptology, with an emphasis on Block Cipher and Authenticated Encryption.

Mohammad Reza Aref received the B.S. degree in 1975 from the University of Tehran, Iran, and the M.S. and Ph.D. degrees in 1976 and 1980, respectively, from Stanford University, Stanford, CA, USA, all in electrical engineering. He returned to Iran in 1980 and was actively engaged in academic affairs. He was a faculty member of Isfahan University of Technology from 1982 to 1995. He has been a Professor of Electrical Engineering at Sharif University of Technology, Tehran, since 1995, and has published more than 290 technical papers in communications, information theory and cryptography in international journals and conferences proceedings. At the same time, during his academic activities, he has been involved in different political positions. First Vice President of I.R. Iran, Vice President of I.R. Iran and Head of Management and Planning Organization, Minister of ICT of I.R. Iran, and Chancellor of University of Tehran, are the most recent ones. His current research interests include areas of Communication Theory, Information Theory, and Cryptography.

Nasour Bagheri is an assistant professor at Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran. He is the author of over 50 articles in information security and cryptology.

Alireza Rahimi is an assistant professor at Faculty of Communication and Information Technology, Imam Hossein University, Tehran, Iran. His interesting is the mathematics of cryptography. He is the author of some articles in mathematics and cryptography.