The ISC Int'l Journal of Information Security

July 2014, Volume 6, Number 2 (pp. 125–139)

http://www.isecure-journal.org

Artemia: A Family of Provably Secure Authenticated Encryption Schemes

Javad Alizadeh [1,2], Mohammad Reza Aref [1], and Nasour Bagheri [3,*]

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

[2] *Faculty and Research Center of Communication and Information Technology, Imam Hossein University, Tehran, Iran*

[3] *Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 8 July 2014
Revised: 12 January 2015
Accepted: 24 January 2015
Published Online: 15 February 2015

Keywords:
Privacy, Authentication, Provable Security, Authenticated Encryption, Artemia.

ABSTRACT

Authenticated encryption schemes establish both privacy and authenticity. This paper specifies a family of the dedicated authenticated encryption schemes, Artemia. It is an online nonce-based authenticated encryption scheme which supports the associated data. Artemia uses the permutation based mode, JHAE, that is provably secure in the ideal permutation model. The scheme does not require the inverse of the permutation in the decryption function, which causes the resource efficiency. Artemia permutations have an efficient and a simple structure and are provably secure against the differential and linear cryptanalysis. In the permutations, MDS recursive layers are used that can be easily implemented in both software and hardware.

1 Introduction

Privacy and authentication are two main goals in information security. In many applications, these security parameters must be established simultaneously. A cryptographic scheme that provides both privacy and authentication is called authenticated encryption (*AE*). 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 one 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 [6]. 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 [15, 16, 19]. Another problem of these modes is the necessity for running the full round block cipher to process each message block which is a 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* [10]. Some dedicated *AE* schemes are ASC-1 [14], ALE [9], AEGIS [21], FIDES [8], CBEAM [17], and APE [5]. A common approach for constructing a dedicated *AE* is to iterate a random permutation or random function, which is considered as a primitive, 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 a random function to be used in the mode.

In this paper, Artemia, an online single-pass nonce-based authenticated encryption scheme is proposed.

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

ISSN: 2008-2045

Table 1. Comparison between Artemia and some known dedicated *AE* schemes

Dedicated *AE*	Provable Security	AD	Online	Nonce Misuse Resistance	Inverse-Freeness of π (or f)	Reference
ASC-1	Yes	No	No	No	Yes	[14]
ALE	No	Yes	Yes	No	Yes	[9]
AEGIS	No	Yes	Yes	No	Yes	[21]
FIDES	No	Yes	Yes	No	Yes	[8]
CBEAM	No	Yes	Yes	No	Yes	[17]
APE	Yes	Yes	Enc only	Yes	No	[5]
Artemia	Yes	Yes	Yes	No	Yes	This paper

The scheme has two variants, Artemia-128 which uses a 128-bit key and Artemia-256 which uses a 256-bit key. It supports the optional associated data. Artemia is a sponge-based [7] scheme which has two main components: the JHAE mode [4] and the permutation *Artemia*. JHAE is a permutation-based *AE* mode based on JH hash function mode [20]. It has provable security up to $O(2^{n/2})$ queries in the ideal permutation model where $2n$ is the length of the permutation.

The permutation *Artemia* has an efficient and a simple structure and is resistant against the differential and linear cryptanalysis. In order to design the permutation, the MDS recursive layers [12, 13, 18] were used that can be easily implemented in both software and hardware.

The Artemia security relies on the usage of nonces. However, it does not allow the reuse of a nonce under the same key. Artemia does not require the inverse of the permutation in the decryption function, this provides resource efficiency.

In Table 1, a comparison is made between Artemia and some other known dedicated *AE* schemes which were presented before the CAESAR (a comparison between Artemia and the CAESAR candidates can be found in [3]).

The paper is structured as follows: Section 2 specifies the design of Artemia. Security goals and analysis of Artemia are presented in Section 3 and design rationale of the scheme is given in Section 4. Finally, the paper is concluded in Section 5.

2 Artemia Specification

This section defines the family of the dedicated authenticated encryption, namely Artemia. It has two variants with different security levels and resource's requirements. Artemia-256 uses a 512-bit permutation and Artemia-128 uses a 256-bit permutation in the JHAE mode.

2.1 Parameters

Artemia has three parameters, *key*, *nonce*, and *tag* and uses an integer n to denote the length of the parameters . The parameters and their length for Artemia-256 and Artemia-128 are summarized in Table 2.

Table 2. Artemia parameters

	lenght of permutation (2n)	length of key (n)	maximum length of nonce (n)	length of tag (n)
Artemia-256	512	256	≤ 256	256
Artemia-128	256	128	≤ 128	128

2.2 Constants

The permutations of $Artemia-512$ and $Artemia-256$ use six constants denoted by C_0 to C_5. These constants are represented in Table 3 and Table 4.

2.3 Conversions

In order to convert a string to another string of different lengths, the little endian conversions is used.

2.4 Specification of JHAE

In this section, we describe JHAE mode [4], also depicted in Figure 1. JHAE is a mode developed from the JH hash function mode and iterates a fixed permutation $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$. It is a nonce-based, single-pass, and an online dedicated *AE* mode that supports the AD.

2.4.1 Encryption and Authentication.

JHAE accepts an n-bit key K, a nonce N of maximum n bits, a message M, and an optional associated data A, then it produces the ciphertext C and authentication tag T. The pseudo code of the JHAE's encryption-authentication is depicted in Algorithm 1. We assume that the input message after padding, is a multiple of

Table 3. The constants of $Artemia - 512$ in hexadecimal

C_0	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0f1e2d3b
C_1	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 4b5a6978 00000000 00000000 00000000 00000000
C_2	00000000 00000000 00000000 00000000 00000000 00000000 00000000 8796a5b4 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
C_3	00000000 00000000 00000000 c3d2e1f0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
C_4	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 2d3c4b5a 00000000
C_5	00000000 00000000 00000000 00000000 00000000 00000000 69788796 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

Table 4. The constants of $Artemia - 256$ in hexadecimal

C_0	00000000 00000000 00000000 00000000 00000000 00000000 00000000 0f1e2d3b
C_1	00000000 00000000 00000000 00000000 00000000 4b5a6978 00000000 00000000
C_2	00000000 00000000 00000000 8796a5b4 00000000 00000000 00000000 00000000
C_3	00000000 c3d2e1f0 00000000 00000000 00000000 00000000 00000000 00000000
C_4	00000000 00000000 00000000 00000000 00000000 00000000 2d3c4b5a 00000000
C_5	00000000 00000000 69788796 00000000 00000000 00000000 00000000 00000000

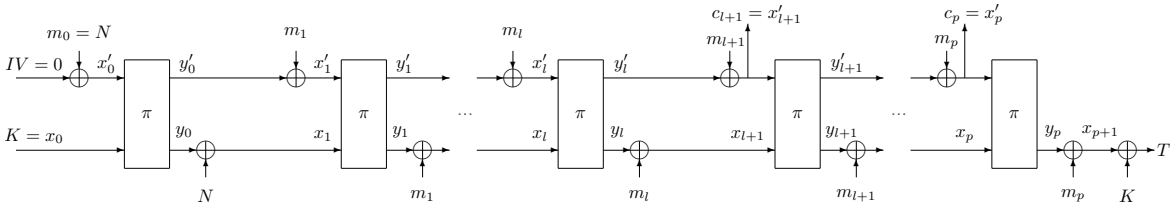


Figure 1. The JHAE mode of operation (the encryption and authentication), where $\text{pad}(A) = m_1 \| m_2 \| \ldots \| m_l$ and $\text{pad}(M) = m_{l+1} \parallel m_{l+2} \parallel \ldots \parallel m_p$

the block size n. The last block of the original message is concatenated by the padding data which is as follows (see Figure 2):

(1) Eight bits are used to represent the length of the nonce (N) in Artemia-256 and nine bits are used to represent the length of the nonce in Artemia-512.
(2) 24 bits are used to represent the length of the associated data (A), e.g., it would be 0^{24} if there is no AD.
(3) 64 bits are used to represent the length of the message (M).
(4) A bit '1' followed by a sequence of '0' is appended 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 3). The padded AD is processed in a way similar to the process of the message block with an exception that ciphertext blocks (c_i), are not produced for the AD blocks.

2.5 Decryption and Verification

JHAE decryption-verification procedure, depicted in Algorithm 2, inputs an n-bit key, K, an maximum n-bit nonce, N, a ciphertext, C, a tag, T, an optional associated data A, and it decrypts the ciphertext to get the message M and tag T'. If $T' = T$, it outputs M else it outputs $\perp$. In the following, we describe the permutations, $Artemia - 512$ and $Artemia - 256$.

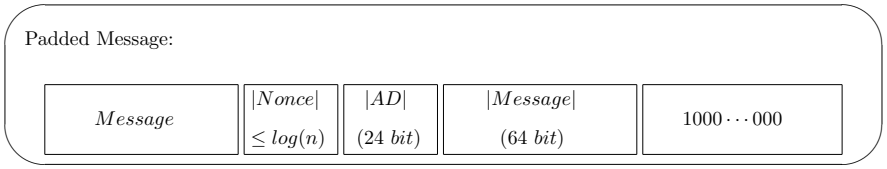


Figure 2. Message padding in JHAE

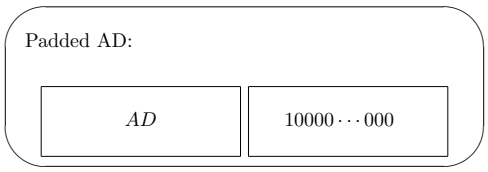


Figure 3. AD padding in JHAE

Algorithm 1 The encryption and authentication pseudo-code of JHAE

Input: Key K of n bits, Nonce N of maximum n bits, Associated data A where $\text{pad}(A) = m_1 \| m_2 \| \ldots \| m_l$ and Message M where $\text{pad}(M) = m_{l+1} \parallel m_{l+2} \parallel \ldots \parallel m_p$. **Output:** Ciphertext C, Tag T.

$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
$\text{pad}(A) \| \text{pad}(M) = m_1 \parallel m_2 \parallel \ldots \parallel m_p$
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
$y'_p \parallel y_p = \pi(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$C = x'_{l+1} \parallel x'_{l+2} \parallel \ldots \parallel x'_p$
$T = x_{p+1} \oplus K$
return (C, T)

2.6 *Artemia* − 512

Artemia − 512 is a 512-bit permutation (*Artemia* − 512 : $\{0,1\}^{512} \rightarrow \{0,1\}^{512}$) which includes the six rounds of $Artemia_{round} - 512 : \{0,1\}^{512} \rightarrow \{0,1\}^{512}$. Now, we explain the round function $Artemia_{round} - 512$.

2.6.1 Specification of $Artemia_{round} - 512$.

$Artemia_{round} - 512$ is depicted in Figure 4 and its pseudo code is represented in Algorithm 3. More precisely, at the beginning of the each round, the 512-bit input state is XORed by a round dependent constant value of the same length. The constant is introduced in § 2.2. Next, the updated state is divided into four words of 128 bits. These 128-bit words are combined with a 4×4 recursive layer ($D1$), and other four words of the 128-bit length is produced. Then, each 128-bit value is passed an SBox layer ($S1$), which is 16 parallel 8×8-bit similar SBoxes and each SBox is applied to a byte of the internal state. Next, each 128-bit word is divided into four words of the 32-bit length and these 32-bit words are combined with a 4×4 recursive layer ($D2$), then other four words of the 32-bit length is produced. In this stage, there are four parallel recursive layers which one processes four words of 32 bits. Then, each 32-bit value is passed an SBox layer ($S2$), which is four parallel 8×8-bit similar SBoxes and each SBox is applied to a byte of the internal state. Given the 16 words of 32 bits, each 32-bit word is divided into four bytes, the bytes are combined with a 4×4 recursive layer ($D3$), and other four bytes are produced. In this stage, there are 16 parallel recursive layers which one process four bytes of the internal state. Finally, each byte passes an SBox ($S3$). In the following, we explain the transformations $D1$, $S1$, $D2$, $S2$, $D3$ and $S3$. $S1$, $S2$ and $S3$ form the confusion layers of $Artemia_{round} - 512$, and $D1$, $D2$, and $D3$ form its diffusion layers.

Algorithm 2 The decryption and verification pseudo-code of JHAE

Input: Key K of n bits, Nonce N of maximum n bits, Associated Data A where $\text{pad}(A) = m_1 \| m_2 \| \ldots \| m_l$, ciphertext $C = c_1 \parallel c_2 \parallel \ldots \parallel c_p$ and Tag T
Output: Message M or $\perp$.

$IV = 0; m_0 = N$
$x'_0 = IV \oplus m_0$; $x_0 = K$
$x'_{l+1} \parallel x'_{l+2} \parallel \ldots \parallel x'_{l+p} = c_1 \parallel c_2 \parallel \ldots \parallel c_p$
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
$y'_p \parallel y_p = \pi(x'_p \parallel x_p)$;
$x_{p+1} = y_p \oplus m_p$
$M = m_{l+1} \parallel m_{l+2} \parallel \ldots \parallel m_p$
$T' = x_{p+1} \oplus K$
if $T' = T$ **then**
  return M
else
  return $\perp$
end if

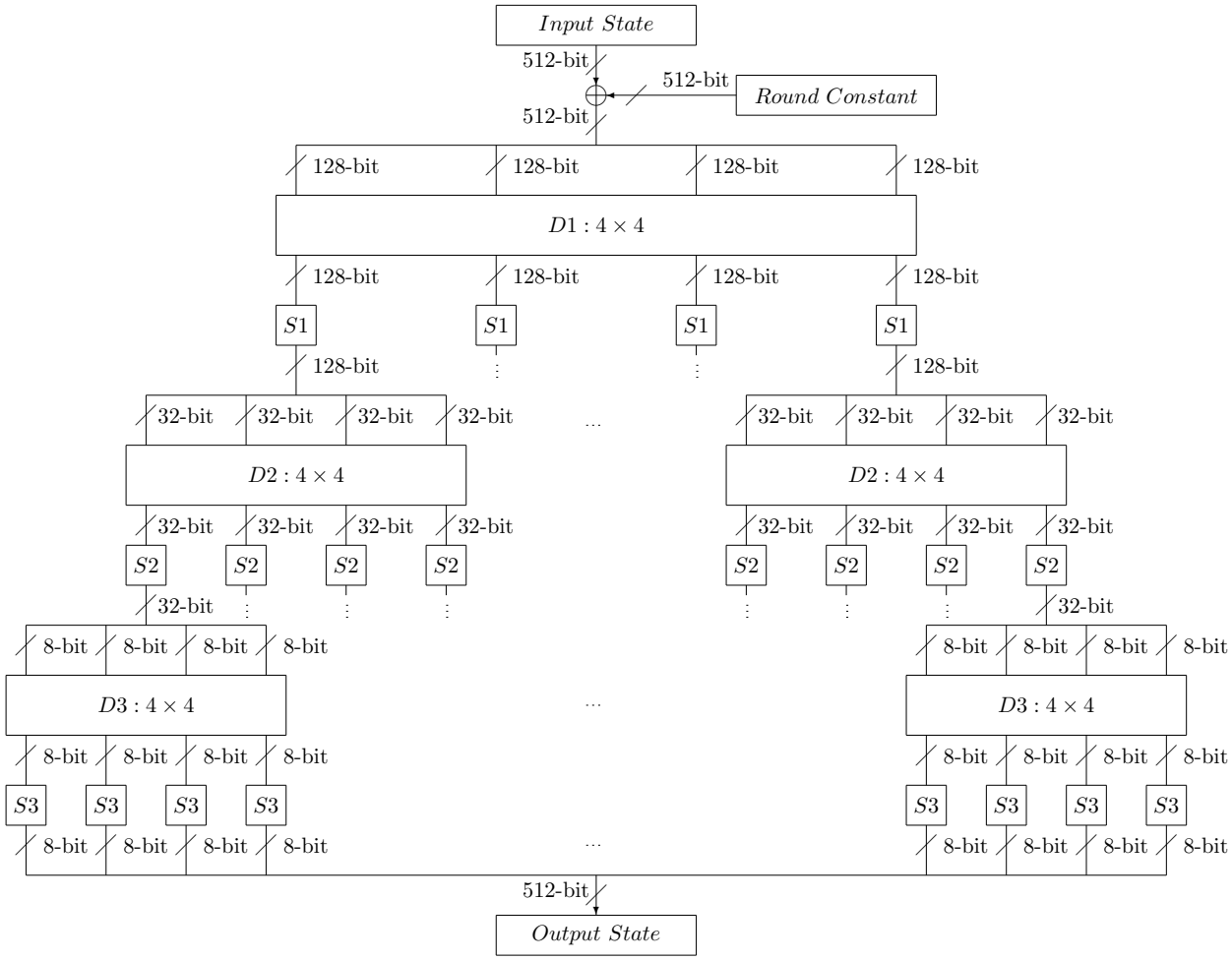


Figure 4. $Artemia_{round} - 512$

2.6.2 Transformations $S1$, $S2$ and $S3$.

All the SBoxes used in the round function are the same and they are identical to the SBox of AES [11]. The lookup table of the SBox is represented in Table 5. For example if $X = \texttt{b2}$ (a byte in hexadecimal notation) is given as the input to the SBox, the output of the SBox would be $y = \texttt{37}$ (in hexadecimal notation).

2.6.3 Transformation $D1$.

$D1$ is a recursive diffusion layer which takes four words of 128 bits of X_0, X_1, X_2 and X_3, and produces four words of 128 bits, Y_0, Y_1, Y_2 and Y_3. The structure of the used diffusion layer was first introduced in [18], and works as follows:

$$\left.\begin{aligned} Y_0 &= X_0 \oplus X_2 \oplus X_3 \oplus L(X_1 \oplus X_3) \\ Y_1 &= X_1 \oplus X_3 \oplus Y_0 \oplus L(X_2 \oplus Y_0) \\ Y_2 &= X_2 \oplus Y_0 \oplus Y_1 \oplus L(X_3 \oplus Y_1) \\ Y_3 &= X_3 \oplus Y_1 \oplus Y_2 \oplus L(Y_0 \oplus Y_2) \end{aligned}\right\} \quad (1)$$

where L is a linear function. If $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$, and $X \oplus L^7(X)$ are invertible, the diffusion layer will be perfect [18] and provides the branch number 5. In addition, if L is an efficient linear function, the diffusion layer would be efficient. In $D1$, we use $L(X) = (X \ll 1) \oplus (X \gg 3)$ that satisfies the given conditions, i.e., $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$, and $X \oplus L^7(X)$ are invertible. Hence, the diffusion layer $D1$ is perfect and efficient and its branch number is 5.

2.6.4 Transformation $D2$.

Similar to $D1$, $D2$ is a recursive diffusion layer which takes four words of 32 bits and produces four words of 32 bits. Its structure is identical to $D1$ with an exception that it works with the 32-bit words. In the case of $D2$, we have $L(X) = (X \ll 1) \oplus (X \gg 3)$. Since $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$, and $X \oplus L^7(X)$ are invertible, $D2$ is a perfect diffusion layer and its branch number is 5.

Algorithm 3 The pseudo-code of $Artemia_{round}-512$

Input: X // a stream of 512-bit length.
Output: Y // a stream of 512-bit length.

C // a constant of 512-bit length from the binary representation of $243F6A888\ldots$;
$X \oplus C = W_3^1 \parallel W_2^1 \parallel W_1^1 \parallel W_0^1$; $//W_i^1 \in \{0,1\}^{128}$; $0 \le i \le 3$.
$W_3^2 \parallel W_2^2 \parallel W_1^2 \parallel W_0^2 = D1(W_3^1 \parallel W_2^1 \parallel W_1^1 \parallel W_0^1)$;
for $i = 0$ to 3 **do**
  $W_i^3 = S1(W_i^2)$;
  $W_i^3 = W_{i,3}^3 \parallel W_{i,2}^3 \parallel W_{i,1}^3 \parallel W_{i,0}^3$; $//W_{i,j}^3 \in \{0,1\}^{32}$; $0 \le j \le 3$.
  $W_i^4 = W_{i,3}^4 \parallel W_{i,2}^4 \parallel W_{i,1}^4 \parallel W_{i,0}^4 = D2(W_{i,3}^3 \parallel W_{i,2}^3 \parallel W_{i,1}^3 \parallel W_{i,0}^3)$;
end for
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    $W_{i,j}^5 = S2(W_{i,j}^4)$;
    $W_{i,j}^5 = W_{i,j,3}^5 \parallel W_{i,j,2}^5 \parallel W_{i,j,1}^5 \parallel W_{i,j,0}^5$; $//W_{i,j,k}^5 \in \{0,1\}^8$; $0 \le k \le 3$.
    $W_{i,j}^6 = W_{i,j,3}^6 \parallel W_{i,j,2}^6 \parallel W_{i,j,1}^6 \parallel W_{i,j,0}^6 = D3(W_{i,j,3}^5 \parallel W_{i,j,2}^5 \parallel W_{i,j,1}^5 \parallel W_{i,j,0}^5)$;
  end for
end for
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    for $k = 0$ to 3 **do**
      $W_{i,j,k}^7 = S3(W_{i,j,k}^6)$;
    end for
  end for
end for
$Y = W_{3,3,3}^7 \parallel W_{3,3,2}^7 \parallel \ldots \parallel W_{0,0,0}^7$;
return Y.

2.6.5 Transformation $D3$.

Similar to $D1$ and $D2$, $D3$ is a recursive diffusion layer given four bytes produces other four bytes. Its structure is identical to $D1$ and $D2$ with two exceptions, that is it works with bytes and uses $L(X) = (X \oplus X \lll 1) \lll 1$. Since $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$, and $X \oplus L^7(X)$ are invertible, $D3$ is a perfect diffusion layer and its branch number is 5.

2.7 $Artemia-256$

$Artemia-256$ is a 256-bit permutation ($Artemia-256 : \{0,1\}^{256} \to \{0,1\}^{256}$) which includes the six rounds of $Artemia_{round}-256 : \{0,1\}^{256} \to \{0,1\}^{256}$. In the rest of this section we describe the round function $Artemia_{round}-256$.

Table 5. The lookup table of the AES SBox

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

2.7.1 Specification of $Artemia_{round} - 256$.

$Artemia_{round} - 256$ is depicted in Figure 5. More precisely, at the beginning of each round, the 256-bit input state is XORed by a round dependent constant value of the same length. The constant is introduced in § 2.2. Next, the updated state is divided into four words of 64 bits. These 64-bit words are combined with a 4×4 recursive layer ($D1$), and other four words of 64 bits are produced. Then, each 64-bit value is passed an SBox layer ($S1$), which is 8 parallel 8×8-bit similar SBoxes and each SBox is applied to a byte of the internal state. Next, each 64-bit word is divided into four words of the 16-bit length and these 16-bit words are combined by a 4×4 recursive layer ($D2$), then other four words of 16-bit length are produced. In this stage, there are four parallel recursive layers which one process four words of 16 bits. Then, each 16-bit value is passed an SBox layer ($S2$), which is two parallel 8×8-bit similar SBoxes and each SBox is applied to a byte of the internal state. Given 16 words of 16 bits, each 16-bit word is divided into two bytes, the bytes are combined by a 2×2 recursive layer ($D3$), and other two bytes are produced. In this stage, there are 16 parallel recursive layers which one processes two bytes of the internal state. Finally, each byte passes an SBox ($S3$). In the following we explain the transformations $D1$, $S1$, $D2$, $S2$, $D3$ and $S3$. $S1$, $S2$ and $S3$ form the confusion layers of $Artemia_{round} - 256$, and $D1$, $D2$ and $D3$ form its diffusion layers.

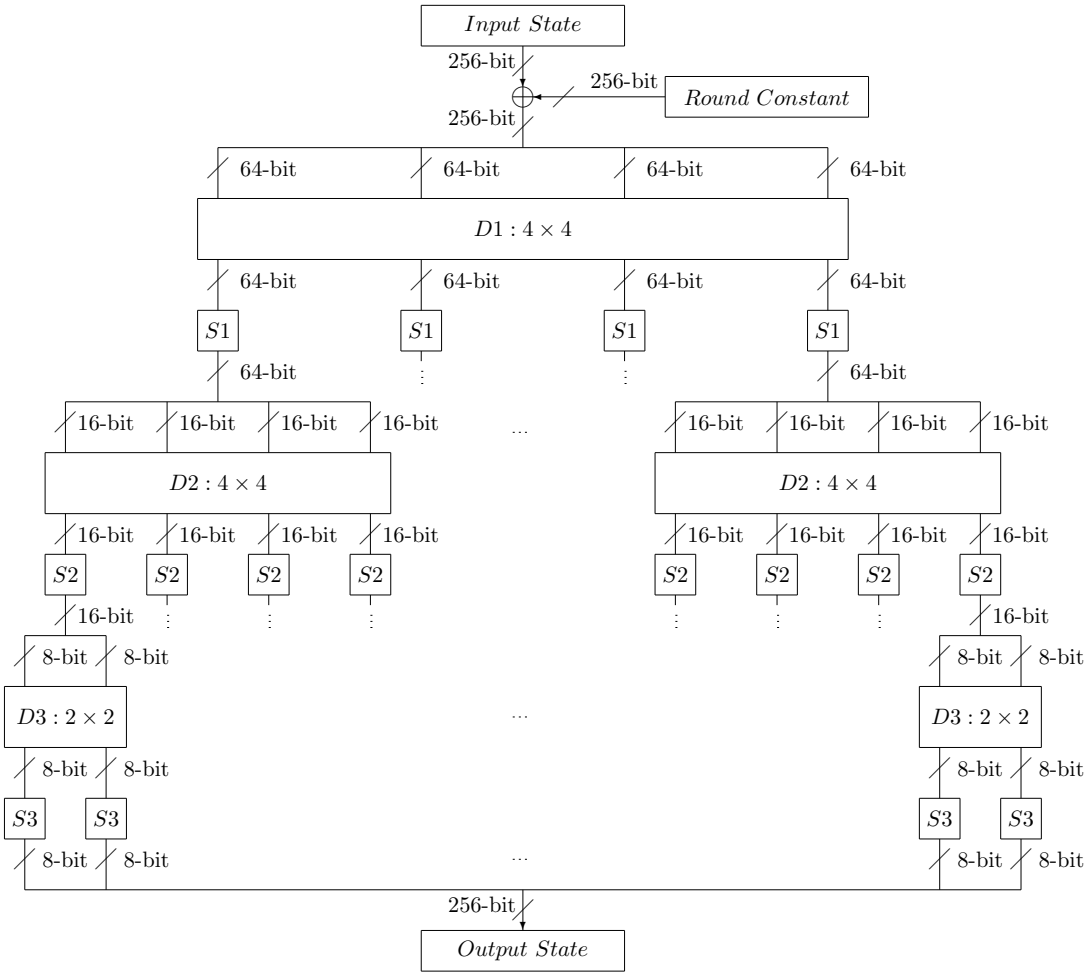


Figure 5. $Artemia_{round} - 256$

The pseudo-code of $Artemia_{round} - 256$ is represented in Algorithm 4.

2.7.2 Transformations $S1$, $S2$ and $S3$.

Similar to $Artemia - 512$, any SBox used in the round function of $Artemia - 256$ is identical to the SBox of AES. The lookup table of the SBox is represented in Table 5.

2.7.3 Transformation $D1$.

$D1$ is a recursive diffusion layer given four words of 64 bits of X_0, X_1, X_2 and X_3, and it produces other four words of 64 bits, Y_0, Y_1, Y_2 and Y_3 as shown in Equation 1. As we discussed about the recursive layers of $Artemia - 512$, if $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$ and $X \oplus L^7(X)$ are invertible, the diffusion layer would be perfect [18] and provides the branch number 5. In $D1$, $L(X) = (X \ll 1) \oplus (X \gg 15)$ satisfying the given conditions is used. Hence, the diffusion layer $D1$ is perfect and its branch number is 5.

2.7.4 Transformation $D2$.

Similar to $D1$, $D2$ is a recursive diffusion layer given four words of 16 bits produces other four words of 16 bits. Its structure is identical to $D1$ with two exceptions that it works with 16-bit words and uses a different L. In the case of $D2$, we have $L(X) = (X \ll 1) \oplus (X \gg 1)$. Since $L(X)$, $X \oplus L(X)$, $X \oplus L^3(X)$, and $X \oplus L^7(X)$ are invertible, $D2$ is a perfect diffusion layer and its branch number is 5.

2.7.5 Transformation $D3$.

Similar to $D1$ and $D2$, $D3$ is also a recursive diffusion layer. However, it is a 2×2 recursive diffusion layer. It is introduced in [18] and works as follows:

$$\left.\begin{aligned} Y_0 &= X_0 \oplus L(X_1) \\ Y_1 &= X_1 \oplus L(Y_0) \end{aligned}\right\} \quad (2)$$

where L is a linear function. It has been shown that if $L(X)$ and $X \oplus L(X)$ are invertible, the diffusion layer is perfect [18]. We use $L(X) = (X \ll 1) \oplus (X \gg$

Algorithm 4 The pseudo-code of $Artemia_{round}-256$

Input: // a stream of 256-bit length.
Output: Y // a stream of 256-bit length.

C //a constant of 256-bit length from the binary representation of $243F6A888\ldots$;
$X \oplus C = W_3^1 \parallel W_2^1 \parallel W_1^1 \parallel W_0^1$; $//W_i^1 \in \{0,1\}^{64}$; $0 \le i \le 3$.
$W_3^2 \parallel W_2^2 \parallel W_1^2 \parallel W_0^2 = D1(W_3^1 \parallel W_2^1 \parallel W_1^1 \parallel W_0^1)$;
for $i = 0$ to 3 **do**
  $W_i^3 = S1(W_i^2)$;
  $W_i^3 = W_{i,3}^3 \parallel W_{i,2}^3 \parallel W_{i,1}^3 \parallel W_{i,0}^3$; $//W_{i,j}^3 \in \{0,1\}^{32}$; $0 \le j \le 3$.
  $W_i^4 = W_{i,3}^4 \parallel W_{i,2}^4 \parallel W_{i,1}^4 \parallel W_{i,0}^4 = D2(W_{i,3}^3 \parallel W_{i,2}^3 \parallel W_{i,1}^3 \parallel W_{i,0}^3)$;
end for
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    $W_{i,j}^5 = S2(W_{i,j}^4)$;
    $W_{i,j}^5 = W_{i,j,3}^5 \parallel W_{i,j,2}^5 \parallel W_{i,j,1}^5 \parallel W_{i,j,0}^5$; $//W_{i,j,k}^5 \in \{0,1\}^8$; $0 \le k \le 3$.
    $W_{i,j}^6 = W_{i,j,3}^6 \parallel W_{i,j,2}^6 \parallel W_{i,j,1}^6 \parallel W_{i,j,0}^6 = D3(W_{i,j,3}^5 \parallel W_{i,j,2}^5 \parallel W_{i,j,1}^5 \parallel W_{i,j,0}^5)$;
  end for
end for
for $i = 0$ to 3 **do**
  for $j = 0$ to 3 **do**
    for $k = 0$ to 3 **do**
      $W_{i,j,k}^7 = S3(W_{i,j,k}^6)$;
    end for
  end for
end for
$Y = W_{3,3,3}^7 \parallel W_{3,3,2}^7 \parallel \ldots \parallel W_{0,0,0}^7$;
return Y

3) (satisfying the conditions) in $D3$. Hence, $D3$ is a perfect diffusion layer and has the branch number 3.

2.8 The Authenticated Encryption Artemia

We define Artemia-256 and Artemia-128 as the two variants of the family of the dedicated authenticated encryption which is named Artemia, as follows.

2.8.1 Artemia-256.

Artemia-256 uses the permutation $Artemia-512$ in the JHAE mode. Its key, AD blocks and message blocks have the length of 256-bit, and it produces the ciphertext blocks and a tag of the 256-bit length.

2.8.2 Artemia-128.

Artemia-128 uses the permutation $Artemia-256$ in the JHAE mode. Its key, AD blocks and message blocks have the length of 128-bit, and it produces the ciphertext blocks and a tag of 128 bits.

Table 6. The security goals of Artemia

Goal	Artemia-256 bits of security	Artemia-128 bits of security
Confidentiality of the secret key	128	64
Confidentiality of the plaintext	128	64
Integrity of the plaintext	128	64
Integrity of the associated data	128	64
Integrity of the nonce	128	64

3 Security Analysis

In this section, security of Artemia is investigated. At first, the security goals of Artemia are presented briefly and then the security of Artemia is described in two subsections: security analysis of JHAE, and security analysis of the permutation *Artemia*.

3.1 Security Goals

The security goals of Artemia are summarized in Table 6. The padding process of Artemia does not use the secret message number. Hence, the bit length of this field is zero. It uses a nonce value as the public message, which is upper bounded by 256 bits for $Artemia-512$ and 128 bits for $Artemia-256$. The only restriction for the nonce value is that reuse of the nonce value under a same key is not allowed. It is unnecessary that the nonce values have equal length (shorter values of the nonce value will be extended to maximum length by appending 0-bit to the left). Hence, the scheme does not provide any integrity or confidentiality if the legitimate user uses a same set (nonce, key) to encrypt two different sets of (plaintext, associated data). In addition, during the decryption, the scheme returns m if the received tag is correct and $\perp$ otherwise.

3.2 Security Analysis of JHAE

In [4] it is shown that JHAE achieves the privacy (indistinguishability under 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 $2n$. The security of JHAE can be summarized in the following two theorems:

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 an 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 $JHAE-E$, π, and π^{-1}, by $\mathcal{A}$.

Proof. The proof of this theorem can be found in [4]. □

Theorem 2. *For any adversary $\mathcal{A}$ that makes σ block queries to $JHAE-E$, π, or π^{-1} in total, JHAE based on an ideal permutation $\pi : \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is (t_A, σ, ϵ)-unforgeable, then $\epsilon \leq \frac{3\sigma^2}{2^n} + \frac{3q}{2^n}$.*

Proof. The proof of this theorem can be found in [4]. □

3.3 Security Analysis of the Permutation *Artemia*

In this section, the security of *Artemia* against differential and linear cryptanalysis is investigated. We show that any 2-round differential or linear characteristic has a minimum of 45 and 35 active SBoxes in *Artemia* − 512 and *Artemia* − 256, respectively. The numbers are a trivial lower bound for the minimum number of active SBoxes. The lower bound can be improved with respect to the diffusion layers of *Artemia* and the linear function that are used in the layers. On the other hand, the differential and linear characteristic of the SBox used in *Artemia* are 2^{-6} and 2^{-4}, respectively. Hence, the probability of any 2-round differential characteristic for *Artemia* − 512 and *Artemia* − 256 are upper bounded by 2^{-270} and 2^{-210}, respectively. Similarly, for any 2-round linear characteristic for *Artemia* − 512 and *Artemia* − 256, the biases are upper bounded by 2^{-180} and 2^{-140}, respectively. By following a similar approach, any 4-round differential characteristic for *Artemia*−512 and *Artemia* − 256 has a probability upper bounded by 2^{-540} and 2^{-420}, respectively and any 4-round linear characteristic for *Artemia* − 512 and *Artemia* − 256 has a bias upper bounded by 2^{-360} and 2^{-280}, respectively. These results are summarized in Table 7.

In the rest of this section, we show the correctness of our claims on the number of active SBoxes for *Artemia* − 512 and *Artemia* − 256.

3.3.1 *Artemia-512*

3.3.2 The Minimum Number of Active SBoxes in Two Rounds.

In Figure 4, assume that a $D3$ recursive layer has been activated.. An active $D3$ guarantees at least five active SBoxes in $S2$ and $S3$. On the other hand, any active SBox in $S2$ comes from an active $D2$ which also guarantees five active SBoxes in $S1$ and $S2$. Hence, each active 128-bit word at the input of $D1$ in the $i-th$ round guarantees at least nine active SBoxes in the $i-th$ round and each active 128-bit word at the output of $D1$ in $i-th$ round guarantese at least nine active SBoxes in the $(i-1)-th$ round. Since the branch number of $D1$ is five, there are at least five active words in the input/output of any active $D1$. Hence, the minimum number of active SBoxes for two rounds of *Artemia* − 512 is 45 (see also Figure 6 in Appendix 6 where the bold line is related to the lower bound). We summarize the minimum number of active SBoxes for two rounds of *Artemia* − 512 in Table 8.

3.3.3 *Artemia-256*

3.3.4 The Minimum Number of Active SBoxes in Two Rounds.

In Figure 5, assume that a $D3$ recursive layer has been activated. An active $D3$ guarantees at least three active SBoxes in $S2$ and $S3$. On the other hand, any active SBox in $S2$ comes from an active $D2$ which also guarantees five active SBoxes in $S1$ and $S2$. Hence, each active 64-bit word at the input of $D1$ in the $i-th$ round guarantees at least nine active SBoxes in the $i-th$ round and each active 64-bit word at the output of $D1$ in $i-th$ round guarantees at least seven active SBoxes in the $(i-1)-th$ round. Since the branch number of $D1$ is five, there are at least five active words in the input/output of any active $D1$. Hence, the minimum number of active SBoxes for two rounds of *Artemia*−256 is 35 (see also Figure 7 in Appendix 6 where the bold line is related to the lower bound). We summarize the minimum number of active SBoxes for two rounds of *Artemia* − 256 in Table 9.

4 Design Rationale

Artemia has two main components: the JHAE mode and the permutation *Artemia*. In order to design each component, we use the publicly known elements. In the following, we give the rationale of the designing each component.

4.1 JHAE

JH [20] is a finalist of the SHA-3 competition and JHAE is a dedicated authenticated encryption mode based on the JH mode. JHAE is a sponge-like mode that uses a permutation and does not need any key schedule. On the other hand, in [4], it has been shown that JHAE is provably secure up to $O(2^{n/2})$. The important researches on JH hash mode done during SHA-3 competition shows that there is no any significant vulnerability in the JH hash mode.

Table 7. The minimum number of active SBoxes and the differential and linear characteristic for *Artemia*

Artemia	# Rounds	# Minimum active SBoxes	Maximum probability of a differential characteristic	Maximum bias of a linear characteristic
Artemia – 512	2	45	2^{-270}	2^{-180}
Artemia – 512	4	90	2^{-540}	2^{-360}
Artemia – 256	2	35	2^{-210}	2^{-140}
Artemia – 256	4	70	2^{-420}	2^{-280}

Table 8. The minimum number of active SBoxes for two rounds of *Artemia* – 512

# active words in the start of the round	# Minimum active SBoxes in the end of the round	# active words in the start of the next round	# Minimum active SBoxes in the end of the next round	# Minimum active SBoxes in two rounds of *Artemia* – 512
1	36	4	9	45
2	27	3	18	45
3	18	2	27	45
4	9	1	36	45

Table 9. The minimum number of active SBoxes for two rounds of *Artemia* – 256

# active words in the start of the round	# Minimum active SBoxes in the end of the round	# active words in the start of the next round	# Minimum active SBoxes in the end of the next round	# Minimum active SBoxes next round in two rounds of *Artemia* – 256
1	28	4	7	35
2	21	3	14	35
3	14	2	21	35
4	7	1	28	35

4.2 The Permutation *Artemia*

The permutation *Artemia* has two main layers: the confusion and diffusion layer. In the confusion layer, the AES SBox having the appropriate characteristics is used. The diffusion layers are developed from the recently introduced recursive diffusion layers [18], that are simple and efficient. In [18], it is shown that these diffusion layers are perfect and provide the maximum branch number.

One can summarize the design rational of Artemia as follows:

- **Security**;
- **Simplicity**;
- **Using the known transformations as its components**;
- **Avoiding a Key Schedule**.

5 Conclusion

We have proposed Artemia, a family of dedicated authenticated encryption (AE) scheme. It is a single-pass nonce-based online scheme, which supports optional associated data. Artemia uses the permutation, *Artemia*, in the JHAE mode.

We showed that the permutation *Artemia* is secure against two most powerful cryptanalysis methods, differential and linear cryptanalysis. On the other hand, the permutation *Artemia* has an efficient and a simple structure. It uses MDS recursive layers that can be easily implemented in both software and hardware. Given that, JHAE is an sponge-based mode which is efficient in both software and hardware and it is provably secure in the ideal permutation model [4], we can claim that Artemia is a family of secure and efficient AE scheme.

We implemented Artemia in software where the results of the implementation can be found in [2] and the test vectors are presented in Appendix 7. A compression between the speed of Artemia and other CAESAR candidates can be found in [1]. It must be noted that the current implementation is a basic

implementation and can be improved. As a future work we improve our software implementation and present an efficient hardware implementation of Artemia.

Acknowledgment

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

References

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

[2] Reference Implementations of the CAESAR Candidates, 2014. http://bench.cr.yp.to/web-impl/amd64-morningstar-crypto_aead.html.

[3] F. Abed, C. Forler, and S. Lucks. Classification of the CAESAR Candidates. Cryptology ePrint Archive, Report 2014/792, 2014. http://eprint.iacr.org/.

[4] J. Alizadeh, M. R. Aref, and N. Bagheri. JHAE: An Authenticated Encryption Mode Based on JH. Cryptology ePrint Archive, Report 2014/193, 2014. http://eprint.iacr.org/.

[5] 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.

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

[7] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Sponge Functions. ECRYPT hash workshop, 2007.

[8] 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.

[9] 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.

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

[11] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[12] J. Guo, T. Peyrin, and A. Poschmann. The PHOTON Family of Lightweight Hash Functions. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

[13] J. Guo, T. Peyrin, A. Poschmann, and M. J. B. Robshaw. The LED Block Cipher. In B. Preneel and T. Takagi, editors, *CHES*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] M. Sajadieh, M. Dakhilalian, H. Mala, and P. Sepehrdad. Recursive diffusion layers for block ciphers and hash functions. In *FSE*, volume 7549 of *Lecture Notes in Computer Science*, pages 385–401. Springer, 2012.

[19] D. Whiting, N. Ferguson, and R. Housley. Counter with CBC-MAC (CCM). *Request for Comments (RFC)*, (3610), 2003.

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

[21] 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.

Appendix

6 The Number of Active SBoxes

7 Test Vectors

The test vectors (in hexadecimal notation) of Artemia-256 and Artemia-128 are given below.

7.1 Test vectors of Artemia-256

7.1.1 a.

Key: 0 *Nonce:* 0 *Message:* 0 *AD:* 0
Ciphertext: 884ec6cf910fd4dfd97c6ca56f71e264f63177495c1d13bff2741227398a8999
Tag: 1190271e3a3aac7d2427f9a5d6a1fde3adbe10e043205a5aa6755b9806653247

7.1.2 b.

Key: 0 *Nonce:* 0 *Message:* 0 *AD:* Empty
Ciphertext: e55ffef4d4ccbd041ec98025eb26cba874a3282c1831aa4bca57519eac039971
Tag: c0a9c7f8f6aac255e6b6a04657235aa90487850aeaf5ff787fe004b8349f17e6

7.1.3 c.

Key: ff *Nonce:* ff *Message:* ff *AD:* ff
Ciphertext: dda879fcfd8b977b01feff3470df656700c7d070840052fd7174ce6e23561136
Tag: 65414aa30abd19039f79ef96accc02d2710e0727bd2469ad1c5ef67de21d762e

7.1.4 d.

Key: ff *Nonce:* ff *Message:* ff *AD:* Empty
Ciphertext: e4064adb94c2f229d205b416a207841463b9a548dd9fb20b90769ffe287356e1
Tag: 00a4f4793851cdaca43ad0c83893ced0978be2fa5a37526d18621628e0771fd2

7.1.5 e.

Key: dbdb *Nonce:* dbdb *Message:* 03bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb5525aeebbbbbbbbb *AD:* 044444444444444444444444444471f36a1243ab77777
Ciphertext: 12b675307357d734ba16daeaeb745ff3797b1c00cca946fbc55d97ead069ef4971d47ce65d6f0e714dfd14d02b2d27742fb52ae4f467410dd5a6d232ff2f73b6
Tag: b45e48ad814639f700412db009087bac447fe549e446c288b49a9b9f347020aa

7.2 Test vectors of Artemia-128

7.2.1 a.

Key: 0 *Nonce:* 0 *Message:* 0 *AD:* 0
Ciphertext: df365dc54f4931a00c0180e5acf3cbfc
Tag: 52dde31b249a0c4b6bb3490cf4833b3f

7.2.2 b.

Key: 0 *Nonce:* 0 *Message:* 0 *AD:* Empty
Ciphertext: 3a1eabbccd33af7fecf1abe9acadc1b6
Tag: 753d513f2dfe79b65f4ebe5800bf0a1c

7.2.3 c.

Key: ff *Nonce:* ff *Message:* ff *AD:* ff
Ciphertext: a5aeb92df745ddaaf764d0510374b147
Tag: 91e29ab5ee55e06b5deecb59038b65b6

7.2.4 d.

Key: ff *Nonce:* ff *Message:* ff *AD:* Empty
Ciphertext: b5583b1d0bbb727b6ad8103e974078f5
Tag: 4f92f91749a91aff0825097319b06652

7.2.5 e.

Key: dbdb *Nonce:* dbdb
Message: 03bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb5525aeebbbbbbbbb
AD: 044444444444444444444444444471f36a1243ab77777
Ciphertext: eb70ad7cbcbd7da09a5063e7abd53fa38fcaeff5ae062a98729824d2d035ff45dcb5ef8c4328bd2814e1969f3e0ab89f
Tag: 56af39a4050397caaff0989284471681

8 Name

We named it *Artemia* because of:

Critical condition of Artemia Urmiana and possibility of extinction[1] .

[1] See http://saveurmia.com/main/2013/01/11/critical-condition-of-artemia-urmiana-and-possibility-of-extinction/

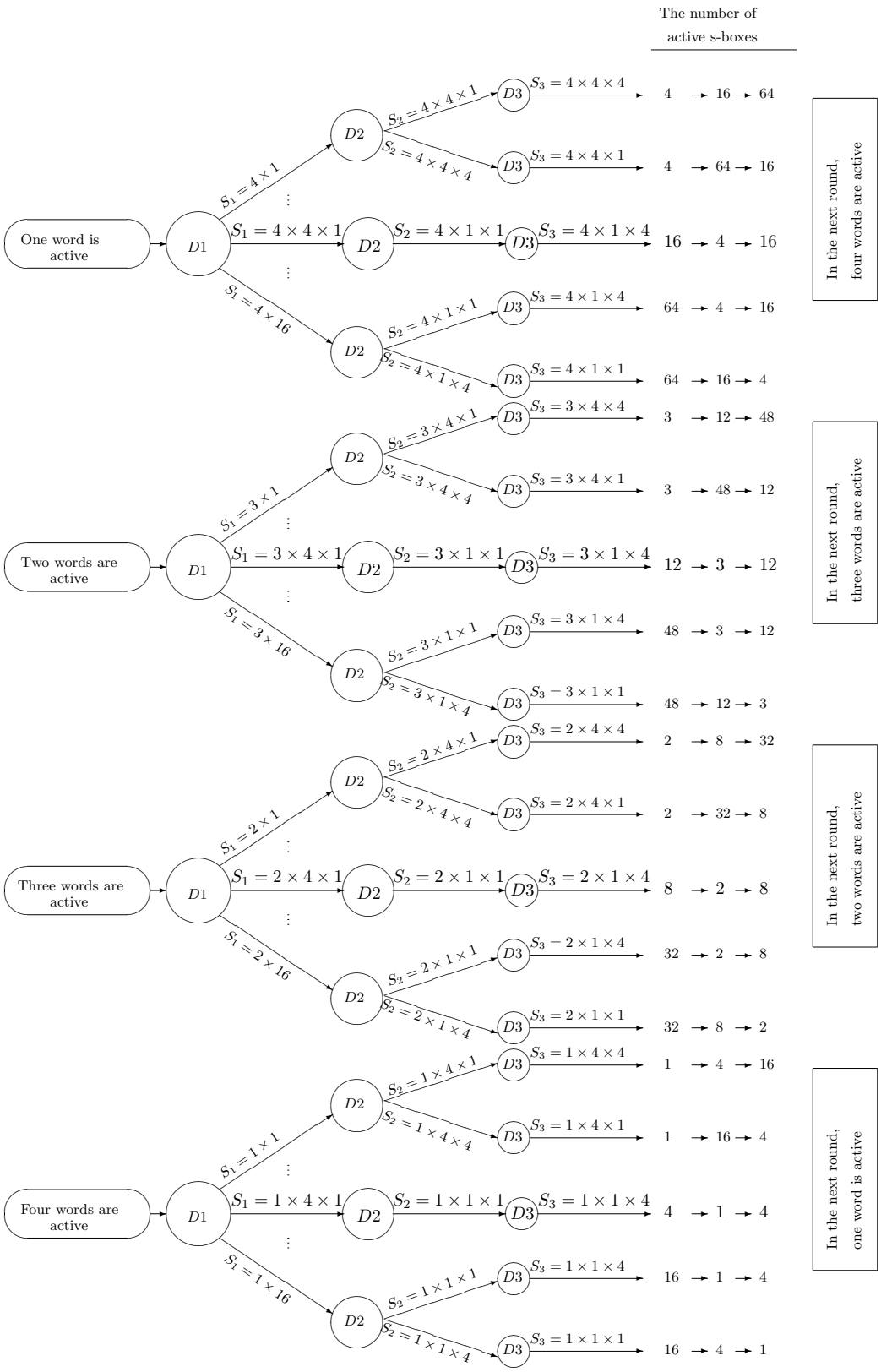


Figure 6. The minimum number of SBoxes in *Artemia* − 512. S_1, S_2, and S_3 are the minimum number of SBoxes in S1, S2, and S3 respectively

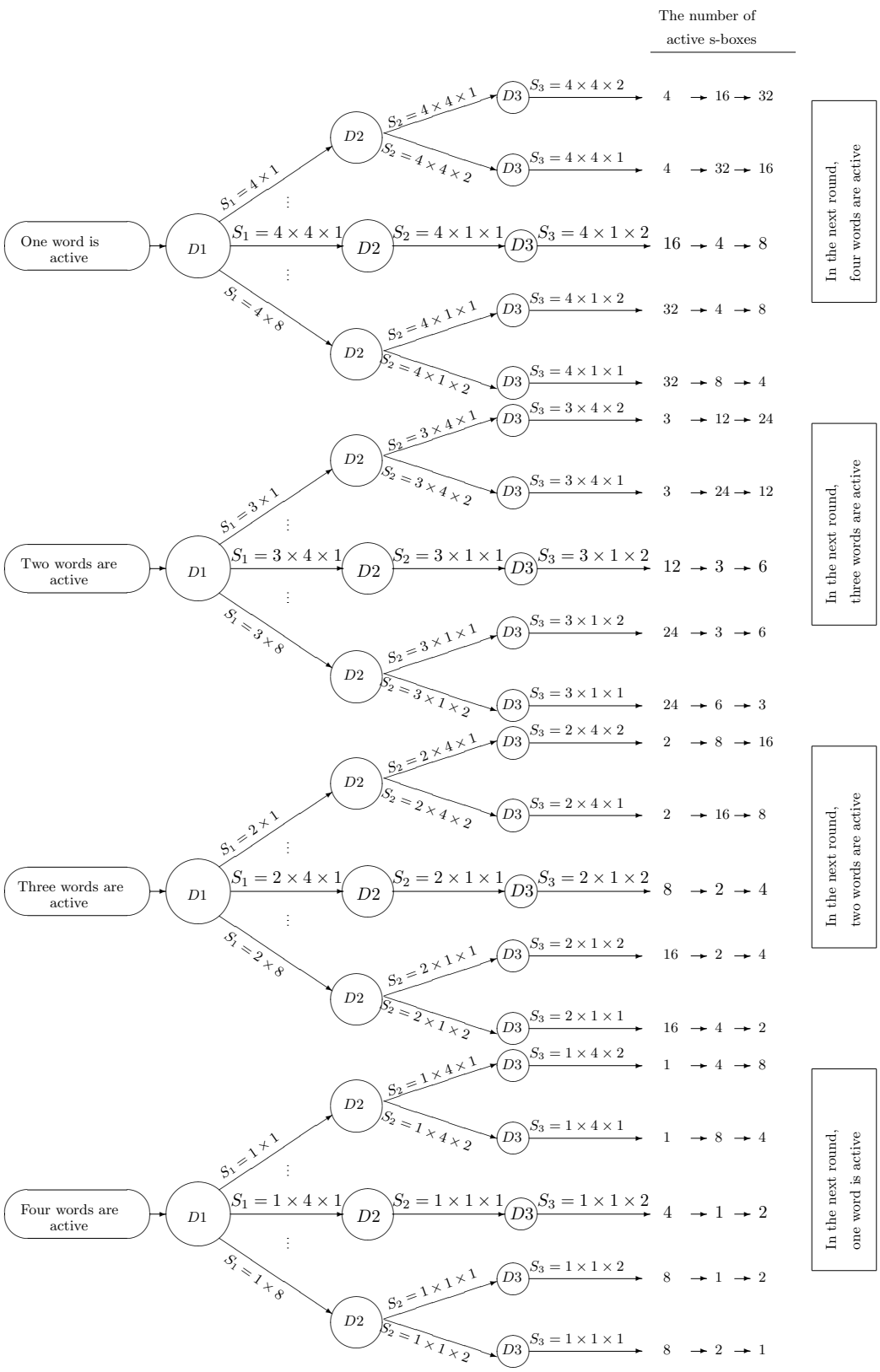
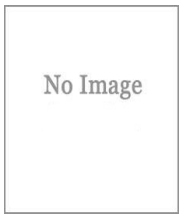


Figure 7. The minimum number of active SBoxes in *Artemia* – 256. S_1, S_2, and S_3 are the minimum number of SBoxes in S1, S2, and S3 respectively



Javad Alizadeh received M.S. degree in Telecommunication in the field of Cryptography from Imam Hoseein 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 Hoseein University. His research interest include symmetric cryptography, 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. 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.