symmetry

MDPI

Article

Authenticated Encryption Based on Chaotic Neural Networks and Duplex Construction

Nabil Abdoun [1], Safwan El Assad [2,*], Thang Manh Hoang [3], Olivier Deforges [4], Rima Assaf [5] and Mohamad Khalil [5]

1 Research and Development Department, SysDICE GmbH, 68259 Mannheim, Germany; nabil.abdoun@sysdice.com
2 Institut d'Électronique et des Technologies Numérique, Polytech Nantes, UMR CNRS 6164, 35042 Nantes, France
3 Vietnam-Japan International Institute for Science of Technology, School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, 1 Dai Co Viet, Hai Ba Trung, Hanoi 10999, Vietnam; thang.hoangmanh@hust.edu.vn
4 Institut d'Électronique et des Technologies Numérique, UMR CNRS 6164, 35708 Rennes, France; olivier.deforges@insa-rennes.fr
5 Faculty of Engineering, Lebanese University, Beirut 90656, Lebanon; rima.assaf@ul.edu.lb (R.A.); mohamad.khalil@ul.edu.lb (M.K.)
* Correspondence: safwan.elassad@univ-nantes.fr; Tel.: +33-676322836

Abstract: In this paper, we propose, implement and analyze an Authenticated Encryption with Associated Data Scheme (AEADS) based on the Modified Duplex Construction (MDC) that contains a chaotic compression function (CCF) based on our chaotic neural network revised (CNNR). Unlike the standard duplex construction (SDC), in the MDC there are two phases: the initialization phase and the duplexing phase, each contain a CNNR formed by a neural network with single layer, and followed by a set of non-linear functions. The MDC is implemented with two variants of width, i.e., 512 and 1024 bits. We tested our proposed scheme against the different cryptanalytic attacks. In fact, we evaluated the key and the message sensitivity, the collision resistance analysis and the diffusion effect. Additionally, we tested our proposed AEADS using the different statistical tests such as NIST, Histogram, chi-square, entropy, and correlation analysis. The experimental results obtained on the security performance of the proposed AEADS system are notable and the proposed system can then be used to protect data and authenticate their sources.

Keywords: authenticated encryption; duplex construction; chaotic neural network; statistical tests; security analysis; computing performance

Citation: Abdoun, N.; El Assad, S.; Manh Hoang, T.; Deforges, O.; Assaf, R.; Khalil, M. Authenticated Encryption Based on Chaotic Neural Networks and Duplex Construction. *Symmetry* **2021**, *13*, 2432. https://doi.org/10.3390/sym13122432

Academic Editor: Giuseppe Grassi

Received: 25 October 2021
Accepted: 8 December 2021
Published: 16 December 2021

1. Introduction

1.1. Research Background

Authenticated encryption (*AE*) is a term used to describe encryption systems that simultaneously protect the confidentiality, integrity, and authenticity of the data transmitted over insecure channels. Many applications and protocols require these forms of security. However, so far, these properties have been designed separately [1–3].

The *AE* schemes are classified into three groups. The first group is a straightforward class called Encrypt-Then-MAC (*ETM*), which is designed and implemented by using both a hash function and a block cipher, e.g., the *CBC-HMAC* scheme [4] that is widespread used in IP security protocol suite *IPSec* and other applications [5]. The second group is composed of *AE* schemes based on block cipher, e.g., *CCM*, *GCM*, *EAX*, and *OCB* [6], and some of these schemes are even standardized or recommended by National Institute of Standards and Technology (*NIST*). The third group is the *AE* schemes based permutation [7], which were proposed and implemented with the appearance of *Sponge* functions, e.g., *ASCON* [8], *PHOTON-Beetle* [9], *Oribatida* [10], etc. The last type is a new research topic, and these three

types of schemes exhibit some useful properties in terms of key agility, software efficiency and hardware implementation [11].

In 2011, Bertoni et al. [12] proposed an *AE* scheme called *SpongeWrap* which is based on the *Duplex* construction [13]. A *Duplex* construction is a *Sponge* function that allows the extraction of certain bits from the intermediate chaining variables for each block under processing (see Figure 1). It is claimed that the security level of the *Duplex* is equivalent to the *Sponge*'s security [12]. With such a construction, many applications can be built, such as reseedable Pseudo Random bit sequence Generator (*PRG*) and efficient *AE* schemes, requiring only one call to the permutation function f per input block [14]. In fact, the *Duplex* construction is composed of two phases: the first one is the Initialization phase while the second one is the Duplexing phase. In the first phase, the Initial Value *IV*, set to 0, is composed of an outer part with the size equal to the bitrate r and an inner part with the size equal to the capacity c. The input message M of arbitrary length L is divided into blocks $M_i, (i \geq 1)$, of maximum length equal to the maximum duplex rate ρ_{max} [15]. The value of ρ_{max} is given by:

$$\rho_{max}(Pad, r) = max\{|M_i| : |M_i| + |Pad[r](|M_i|)| \leq r\} \quad (1)$$

where $|M_i|$ is the length of the block message $M_i, (i \geq 1)$, *Pad* is a function that pads each block message M_i to obtain $|M_i|$ equal to r, and $Pad[r](|M_i|) = (r - |M_i|)$-bit padded to the message block M_i.

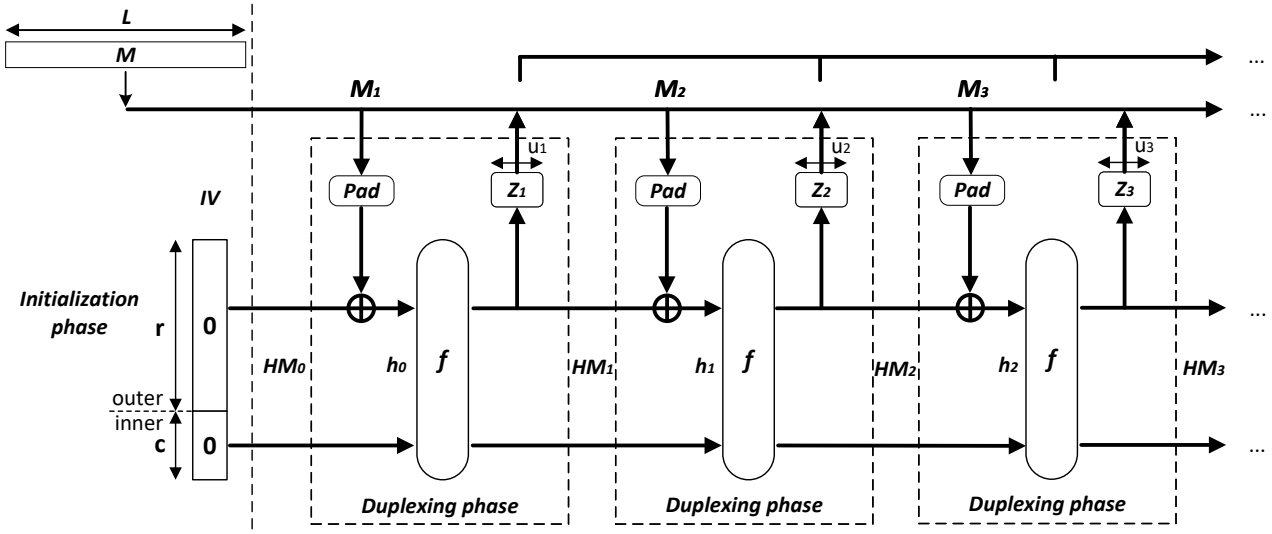
The padding scheme should be injective, reversible, and must ensure that the last block is non-zero [16]. Normally, the simplest padding sequence is equal to multiples of 10 or 10*. In case $\rho_{max}(Pad, r)$ is equal to $r - 1$, the $|Pad[r](|M_i|)|$ is equal to 1 [17,18].

In the second phase, each M_i is padded using the function *Pad* and called one time. This ensures that every padded block is non-zero. Unlike the *Sponge* function, the *Duplex* construction takes an input string for each call, and returns an output string Z_i of length u_i that depends on all previous inputs [7].

Similar to the *Sponge* function, the *Duplex* construction can be converted to a cryptographic keyed hash algorithm by using one of the three methods shown in [19,20]:

1. *FKS* (Full-State Keyed Sponge);
2. *IKS* (Inner keyed-Sponge);
3. *OKS* (Outer keyed-Sponge).

In our proposed *CNN-Duplex* schemes for the *AEAD* applications, the (*FKS*) is inherently included.



M: Message, ***L***: Length, ***IV***: Initial Value, ***Pad***: Padding, ***f***: function, ***r***: rate, ***c***: capacity.

Figure 1. The *Duplex* construction.

1.2. Research Significance

The specificity of our proposed *AEADS* system is in the use of a chaotic compression function instead of a permutation function. It is well known that non-linear dynamic discrete time systems (chaotic maps) can generate chaos and the attractor (signature) of each chaotic map is characterized by the presence of symmetrical patterns [21]. Two chaotic maps, namely the Skew-tent (*ST*) map and *PWLCM* map used in the proposed chaotic compression function, have symmetrical mapping related to their parameters.

This paper is organized into the following sections: Section 2 provides a short description of mostly related works; in Section 3, we describe in details the proposed *AEADS* based on the *MDS-CNNR* structure that comes up to solve existing issues in most of the works completed in the literature. Section 4 presents and discusses the results with different security metrics. Section 5 analyzes the features of the *AEAD* schemes covered in this work and compares the performance with our proposed *AEADS*. Finally, Section 6 concludes this work and suggests future research directions.

2. Related Works

Many methods exist to construct different kinds of *AE* schemes [22]. The simple method is to immediately use the *AE* modes already existed, recommended and standardized by *NIST*. A more complicated way that forms a challenging way is to design a completely new *AE* mode. Note that in the third round of the *CAESAR* competition, most candidates proposed, designed, and implemented new *AE* modes. In fact, Berton et al. designed in 2016 a new *AE* mode adopted by the *Ketje* scheme, an authenticated encryption scheme based on *Keccak-p*, called *MonkeyWrap* [23]. Actually, it is very similar to the *SpongeWrap* method proposed by Bertoni et al. in 2011 [7]. The major difference between these two schemes is that *MonkeyWrap* adopts, during its whole process lifecycle, *MonkeyDuplex* permutations. In fact, *MonkeyWrap* uses two mixing layers to obtain the ciphertexts; the first one is applied on the input data, while the second one is applied on the output data. Additionally, between these two mixing layers, *MonkeyWrap* XORs the message number with the internal data processed by the system. On the other hand, *MonkeyWrap* iterates the obtained message blocks several times in order to calculate the authenticated tag. Furthermore, Bertoni et al. [24] designed in 2015 another *AE* scheme based on a *Sponge* function, called *Motorist*, that was adopted by the *Keyak* scheme. First, it encrypts the message blocks and its number with a unique private vector in order to obtain the authenticated tag and the corresponding cipher blocks. *Motorist* applies the same procedure to all message blocks in parallel, and that is performed by supporting more than one duplex operation at the same time. In 2016, Jean et al. designed an *AE* scheme, called *Deoxys*, based on *Deoxys-BC* [25]. *Deoxys* is an efficient scheme for small messages (few dozen bytes), which is essentially important for various lightweight applications [26]. In this proposed scheme, Jean et al. combines the inputs of authenticated ciphers to obtain the authenticated tags and ciphertexts. Additionally, with the same techniques as the Advanced Encryption Standard algorithm, it can resist side-channel attacks [27].

Moreover, Dobraunig et al. [8] proposed in 2016 an *AE* scheme, intuitively defined on words with 64-bit length, called *Ascon*. The words used in this scheme are only composed from the bitwise Boolean functions like *OR*, *AND*, *XOR*, *ROT*, and *NOT* [28]. The permutation of *Ascon* scheme is similar to this used in *MonkeyDuplex* construction. The authenticated tag and ciphertexts are produced after processing input data including the nonce, secret key, *AD*, and message blocks. For both software and hardware implementation, *Ascon* provides, in terms of speed and size, lightweight characteristics and has also good performance on both implementation. Hence, it is highly recommended in the Internet of Things (*IoT*) use case, where various lightweight computing devices communicate with a back-end server using different protocols. In 2019, Bao et al. [9] designed an authenticated encryption and hash family called *PHOTON-Beetle*, that adopts the *Sponge*-based construction mode *Beetle* with permutation value equal to P_{256} (it is also used for the *PHOTON* hash [29]). In fact, *PHOTON-Beetle* can be categorized into two classes based on its functionalities:

PHOTON-Beetle-AEAD which is a family of authenticated encryption, and *PHOTON-Beetle-Hash* which is a family of hash functions. These two families are parameterized by the rate of message absorption r. The main innovation behind *Beetle Sponge* mode is the combination of two values, i.e., the ciphertext block and the feedback of the permutation output, to generate the next permutation input. Bhattacharjee et al. proposed *Oribatida*, a family of lightweight *AE* schemes, where its design is based on the known duplex construction mode [10]. At its core, *Oribatida*, a variant of the *MonkeyWrap AE* mode, extended by a ciphertext masking that boosts the security, inherits the minimal security guarantees of the well-known duplex construction mode. Moreover, all members of this proposed family are expected to provide 128-bit security for hashing and encryption.

3. Description of the Proposed *AEADS* Based on the *Keyed MDS-CNNR* Structure

The architecture of the proposed *AEADS* based on the *Keyed MDS-CNNR* structure is composed of two phases: Initialization phase and Duplexing phase (see Figure 2).

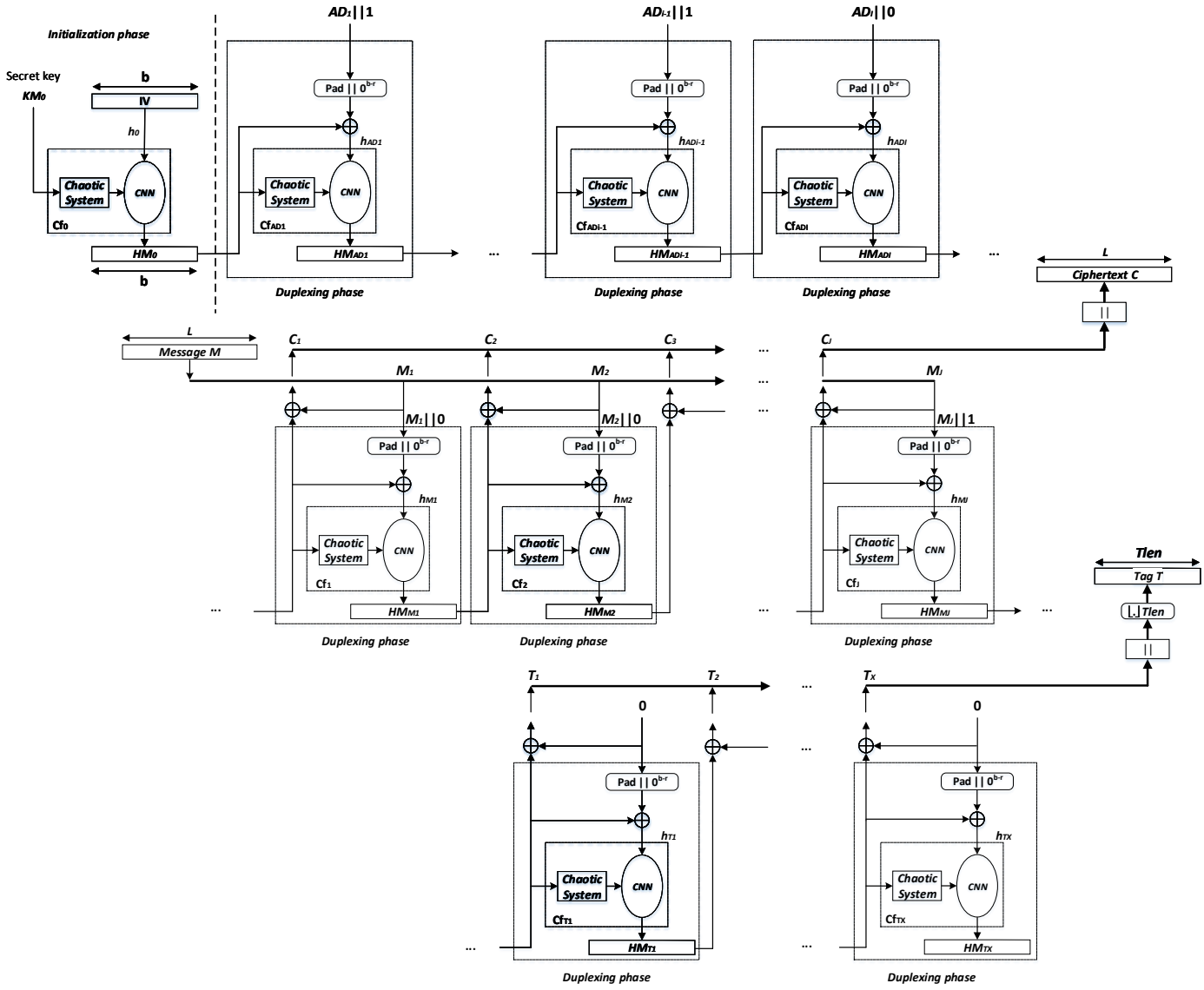


Figure 2. Structure of the proposed *CNN-Duplex* for *AEADS*—Encryption process.

The initialization phase uses the Initial Value *IV* of b-bit length to set the value of h_0. Additionally, the secret key of 160-bit length is used to supply the Chaotic System

(*CS*), $KM_0 = K$, formed by a recursive cell of order one which contains a discrete Skew tent map [30].

Then, the Chaotic compression function Cf_0, composed *CS* and Chaotic Neural Network (*CNN*), takes KM_0 and h_0 to produce the first chaining variable HM_0 of b-bit length.

For this phase, we define the function *CNND.initialize*() for our proposed *CNN-Duplex* construction. This function is explained by Algorithm 1.

Algorithm 1 The initialization function

Require: $r < b$, $\rho_{max}(Pad, r) > 0$
Interface: $HM_0 = CNND.initialize(K, IV)$ *with* $K \in \mathbb{Z}_2^{|K|}$ *and* $IV \in \mathbb{Z}_2^b$
$KM_0 = K$
$h_0 = IV$
$HM_0 = Cf_0(KM_0, h_0)$
Return HM_0.

Where *CNND* is a *CNN-Duplex* object of the *CNNDuplex*[*Cf*, *Pad*, *r*] function.

Notice that for the input data of the other cells, we choose two values of the bitrate r and the capacity c: the first choice is r = 256 bits and c = 256 bits (b = 512 bits = 64 bytes), and the second choice is r = 512 bits and c = 512 bits (b = 1024 bits = 128 bytes).

The duplexing phase takes an input string S and returns an output of b bits. In Algorithm 2, we define the *CNND.duplexing*() function for our proposed *CNN-Duplex* construction.

Algorithm 2 The duplexing function

Require: $r < b$, $\rho_{max}(Pad, r) > 0$
Interface: $HM = CNND.duplexing(S, b)$ *with* $S \in \cup_{l=0}^{\rho_{max}(Pad,r)} \mathbb{Z}_2^l$, *and* $HM \in \mathbb{Z}_2^b$
$KM = \lfloor HM \rfloor_{|KM|}$
$h = HM \oplus \{(S||bf||Pad[r](|S||bf|))||0^{b-r}\}$
$HM = Cf(KM, h)$
Return HM.

Where, l is the length of input string S; bf is the bit-frame equal to 0 or 1; and the input string S of (r-2)-bit length can be *AD*, *M*, or 0. Notice that the length of the last block can be arbitrary. Additionally, the calls where S is an empty string are referred to as blank calls, while the calls with b = 0 are referred to as mute calls.

In this phase, the sub-keys of the *CS* of size $|KM|$ = 128 bits are extracted from the intermediate chaining variable *HM* of size $|HM| = b$ bits ($KM = LSB(HM)$). These *HM* are XORed with $(S||bf||Pad[r](|S||bf|))||0^{b-r})$ to produce the input h of b-bit length used by the *CNN* function. The output of *CS* supplies the parameters of *CNN*.

The authenticated encryption with associated data function *CNNDuplex-AEAD* takes the following elements as inputs:

1. An Initial Value *IV*, and a secret key *K*;
2. An Associated Data (*AD*) that will be authenticated but not encrypted;
3. A message *M* that will be both authenticated and encrypted.

In general, the Associated Data *AD* and the message *M*, when they are available, are divided into multiple blocks $AD_i, (i = 1, \ldots, I)$ and $M_j, (j = 1, \ldots, J)$, respectively. It must be noted that the blocks of *AD* ($AD_i; i = 1, \ldots, I$) are concatenated by a *bf* equal to 1, except for the last block, wherein its *bf* is equal to 0. Moreover, the blocks of *M* ($M_j; j = 1, \ldots, J$) are concatenated by a *bf* equal to 0, except for the last block, wherein its *bf* is equal to 1. The bit-frame *bf* is used for two goals: first, to ensure that both the generated key stream and the obtained authentication tag blocks are in two separate domains; second, to assure that the *AD* and *M* input blocks can be recovered later from the input sequence of the duplex construction [12]. After the absorption of $AD||bf$, the scheme absorbs $M||bf$

and, then generates the ciphertext C, block by block: $C_j = M_j \oplus \lfloor HM \rfloor_{|M_j|}$ of length $|C_j| = |M_j|$; and HM is the response of the *CNND* to the previous *CNND.duplexing*() call. If the needed authentication tag T's length (*Tlen*) is greater than r-2, then the duplexing phase enters in the tag generation phase, where all input strings are equal to 0. The obtained ciphertext C and the authentication tag T will be sent to the receiver.

For each *CNNDAEAD.encrypt*(AD, M, b) request, the latter transmits the blocks of the AD and the message M to the *CNND.duplexing*(S, b). A formal definition and description of *CNNDuplex-AEAD* encryption function is provided in Algorithm 3.

Algorithm 3 The *Authenticated Encryption* function $CNNDuplex - AEAD[Cf, Pad, r, \rho]$

Require: $\rho \leq \rho_{max}(Pad, r) - 1$
Require: $CNND = CNNDuplex[Cf, Pad, r]$
This algorithm treats AD, M, C instances equal to the empty string as a single (empty) block
Interface: $CNNDAEAD.initilalize(K, IV)$ *with* $K \in \mathbb{Z}_2^{|K|}$ *and* $IV \in \mathbb{Z}_2^b$
$HM_0 = CNND.initialize(K, IV)$
Interface: $(C, T) = CNNDAEAD.encrypt(AD, M, b)$
with $AD, M \in \mathbb{Z}_2^*, b \geq 0, C \in \mathbb{Z}_2^{|M|}$ *and* $T \in \mathbb{Z}_2^{Tlen}$
Let $AD = AD_1||AD_2||\ldots||AD_I$ *with* $|AD_i| = \rho$ *for* $i < I$, $|AD_I| \leq \rho$ *and* $|AD_I| > 0$ *if* $I > 0$
Let $M = M_1||M_2||\ldots||M_J$ *with* $|M_j| = \rho$ *for* $j < J$, $|M_J| \leq \rho$ *and* $|M_J| > 0$ *if* $J > 0$
for $i = 1$ *to* $I - 1$ **do**
  $CNND.duplexing(AD_i||1, 0)$
end for
$HM = CNND.duplexing(AD_I||0, |M_1|)$
$C = M_1 \oplus \lfloor HM \rfloor_{|M_1|}$
for $j = 1$ *to* $J - 1$ **do**
  $HM = CNND.duplexing(M_j||0, |M_{j+1}|)$
  $C = C||(M_{j+1} \oplus \lfloor HM \rfloor_{|M_{j+1}|})$
end for
$HM = CNND.duplexing(M_J||1, \rho)$
while $|HM| < Tlen$ **do**
  $HM = HM||CNND.duplexing(0, \rho)$
end while
$T = \lfloor HM \rfloor_{Tlen}$
Return (C, T).

Where, *CNNDAEAD* is an instance of the authenticated encryption function *CNNDuplex-AEAD*[*Cf*, *Pad*, r, ρ].

The structure of the decryption process is shown in Figure 3. It takes the following elements as inputs:

1. The same Initial Value IV and the same secret key K taken from the encryption process;
2. The same Associated Data AD used in the encryption process;
3. The ciphertext C to be decrypted;
4. The authentication tag T used to check the integrity of the Associated Data AD and the authenticity of the message M.

As observed from Figure 3, the decryption process is similar to the encryption process by exchanging message $M_j, (j = 1, \ldots, J)$ and ciphertext $C_j, (j = 1, \ldots, J)$ blocks. This means that $M_j = C_j \oplus \lfloor HM \rfloor_{|C_j|}$, where HM is the response of *CNND* to the previous *CNND.duplexing*() call. If the calculated tag T' is equal to the received authentication tag T, then the original message M is delivered. Otherwise, an *Error* message is delivered.

For each *CNNDAEAD.decrypt*(AD, C, T) request, the latter transmits the blocks of the AD and the retrieved message M from C to *CNND.duplexing*(S, b). A formal definition and description of the *CNNDuplex-AEAD* decryption process is provided in Algorithm 4.

Algorithm 4 The *Authenticated Encryption* function $CNNDuplex - AEAD[Cf, Pad, r, \rho]$

Require: $\rho \leq \rho_{max}(Pad, r) - 1$
Require: $CNND = CNNDuplex[Cf, Pad, r]$
This algorithm treats AD, M, C instances equal to the empty string as a single (empty) block
Interface: $CNNDAEAD.initilalize(K, IV)$ *with* $K \in \mathbb{Z}_2^{|K|}$ *and* $IV \in \mathbb{Z}_2^b$
$HM_0 = CNND.initialize(K, IV)$
Interface: $M = CNNDAEAD.decrypt(AD, C, T)$ *with*
$AD, C \in \mathbb{Z}_2^*, T \in \mathbb{Z}_2^{Tlen}$ *and* $M \in \mathbb{Z}_2^{|C|} \cup \{error\}$
Let $AD = AD_1||AD_2||\ldots||AD_I$ *with* $|AD_i| = \rho$ *for* $i < I$, $|AD_I| \leq \rho$ *and* $|AD_I| > 0$ *if* $I > 0$
Let $C = C_1||C_2||\ldots||C_J$ *with* $|C_j| = \rho$ *for* $j < J$, $|C_J| \leq \rho$ *and* $|C_J| > 0$ *if* $J > 0$
Let $T = T_1||T_2||\ldots||T_X$ *with* $|T_x| = \rho$ *for* $x < X$, $|T_X| \leq \rho$ *and* $|T_X| > 0$ *if* $X > 0$
for $i = 1$ *to* $I - 1$ **do**
  $CNND.duplexing(AD_i||1, 0)$
end for
$HM = CNND.duplexing(AD_I||0, |C_1|)$
$M_1 = C_1 \oplus \lfloor HM \rfloor_{|C_1|}$
for $j = 1$ *to* $J - 1$ **do**
  $HM = CNND.duplexing(M_j||0, |C_{j+1}|)$
  $M_{j+1} = C_{j+1} \oplus \lfloor HM \rfloor_{|C_{j+1}|}$
end for
$HM = CNND.duplexing(M_J||1, \rho)$
while $|HM| < T'len$ **do**
  $HM = HM||CNND.duplexing(0, \rho)$
end while
$T' = \lfloor HM \rfloor_{Tlen}$
if $T = T'$ **then**
  Return $M_1||M_2||\ldots||M_J$.
else
  Return Error.
end if

Note that the AD plays the same role as the header of the message in all network applications [31]. The role of the IV is similar to that used in the stream cipher.

The structure of the proposed Chaotic function Cf is shown in Figure 4. It contains a CS and a single layer CNN, followed by a set of non-linear (NL) functions. The input layer of CNN is composed of four neurons. The activation function of each neuron is composed of the Discrete Skew Tent map (*DSTmap)* and the Discrete Piecewise Linear Chaotic map (*DPWLCmap*). The CS produces a Key Stream KS (all necessary samples) in order to supply the two layers, as represented in Equation (2):

$$KS = \{QI, BI, WI, WO\} \tag{2}$$

The size of KS is represented in the following equation:

$$|KS| = |QI| + |BI| + |WI| + |WO| \tag{3}$$

For the first choice (r = 256 bits, c = 256 bits), each input neuron has four input data and three 32-bit parameters BI_k $QI_{k,1}$, and $QI_{k,2}$ (see Figure 5). BI_k is the bias of the neurons at the input layer while $QI_{k,1}$ and $QI_{k,2}$ are the control parameters for the activation function of *DSTmap* and *DPWLCmap*, respectively. Indeed, for this case, the necessary sizes of 32-bit parameters are calculated as follows: $|QI|$ = 8 samples, $|BI|$ = 4 samples, $|WI|$ = 16 samples, $|WO|$ = 4 samples, and so the total size of $|KS|$ is equal to 32 samples. For the second choice (r = 512 bits, c = 512 bits), each input neuron has eight input data and three 32-bit parameters BI_k, $QI_{k,1}$, and $QI_{k,2}$ (see Figure 6). Indeed, for this case, the necessary

sizes of 32-bit parameters are calculated as follow: $|QI|$ = 8 samples, $|BI|$ = 4 samples, $|WI|$ = 32 samples, $|WO|$ = 4 samples, and so the total size of $|KS|$ is equal to 48 samples.

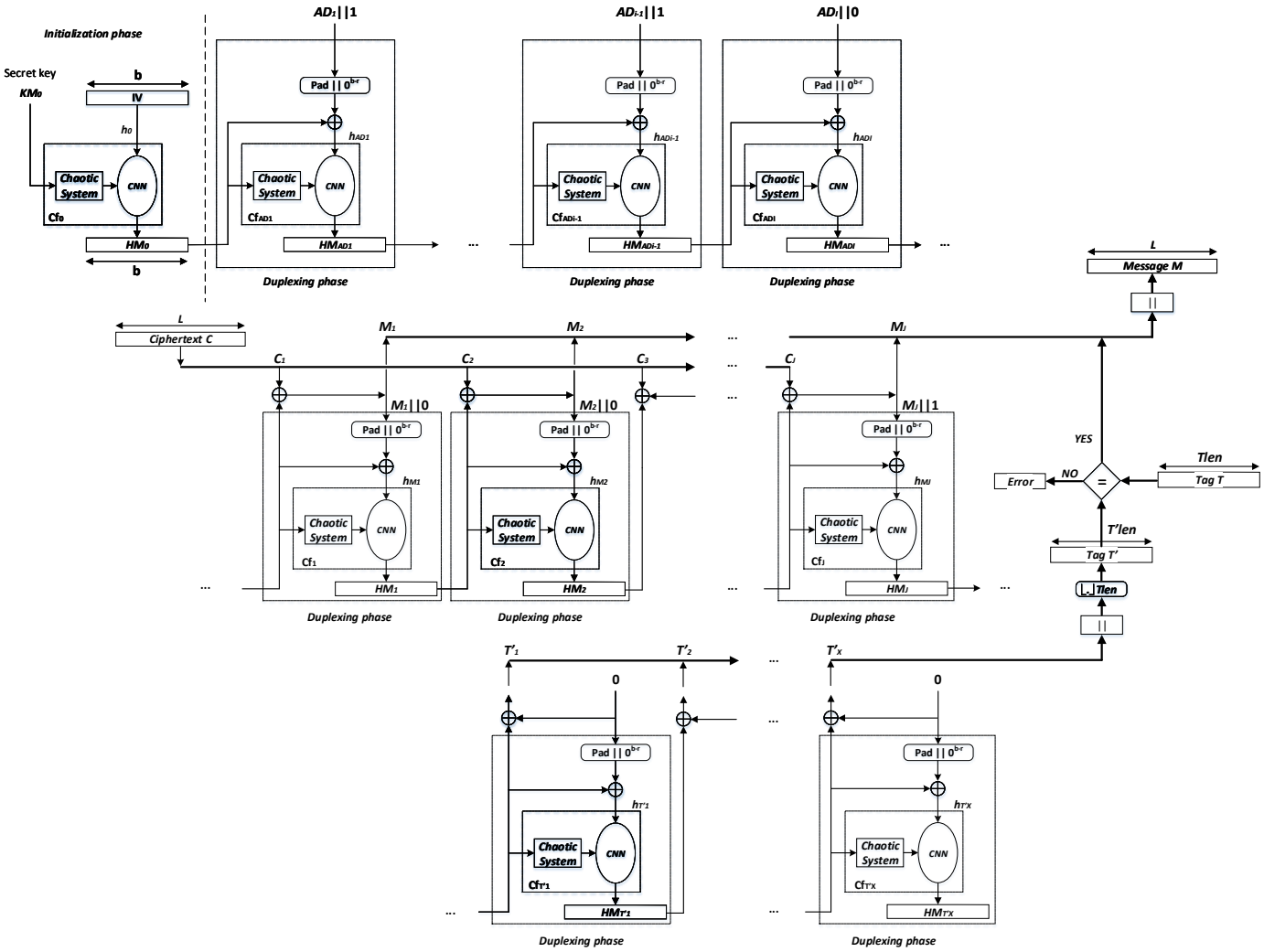


Figure 3. Structure of the proposed *CNN-Duplex* for *AEADS*—Decryption process.

For the first choice, each $h_S, (S = AD, M, T)$ at the input layer is formed by $P_j, (j = 4k, \ldots, 4k+3)$. The first two input blocks $P_j, (j = 4k, 4k+1)$, weighted by the appropriate input weights $WI_j, (j = 4k, 4k+1)$, are both added with the input bias BI_k (weighted by 1) to construct the input of the first *DSTmap* function. The second two input blocks $P_j, (j = 4k+2, 4k+3)$, weighted by the appropriate input weights $WI_j, (j = 4k+2, 4k+3)$, are both added with the same input bias BI_k to construct the input of the second *DPWLCmap* function.

For the second choice, each $h_S, (S = AD, M, T)$ at the input layer is formed by $P_j, (j = 8k, \ldots, 8k+7)$. The first four input blocks $P_j, (j = 8k, \ldots, 8k+3)$, weighted by the appropriate input weights $WI_j, (j = 8k, \ldots, 8k+3)$, are all added with the input bias BI_k (weighted by 1) to construct the input of the first *DSTmap* function. The second four input blocks $P_j, (j = 8k+4, \ldots, 8k+7)$, weighted by the appropriate input weights $WI_j, (j = 8k+4, \ldots, 8k+7)$, are all added with the same input bias BI_k to construct the input of the second *DPWLCmap* function. All inputs P_j are 32-bit samples (integer values),

and the all input biases BI_k are necessary in case having a null input message. The *DSTmap* and the *DPWLCmap* are represented by Equations (4) and (5), respectively.

$$KSs(n) = DSTmap(KSs(n-1), Q1) = \begin{cases} 2^N \times \frac{KSs(n-1)}{Q1} & if\ 0 < KSs(n-1) < Q1 \\ 2^N - 1 & if\ KSs(n-1) = Q1 \\ 2^N \times \frac{2^N - KSs(n-1)}{2^N - Q1} & if\ Q1 < KSs(n-1) < 2^N \end{cases} \quad (4)$$

where, N is the finite precision equal to 32 bits length; and $Q1$ represents the control parameter of the *DSTmap*. *KSs(n)* and *KSs(n-1)* are the outputs of *DSTmap* at the nth and $(n-1)$th iterations, respectively. $Q1$, *KSs(n)*, and *KSs(n-1)* range between 1 and $2^N - 1$.

$$KSp(n) = DPWLCmap(KSp(n-1), Q2) = \begin{cases} 2^N \times \frac{KSp(n-1)}{Q2} & if\ 0 < KSp(n-1) \le Q2 \\ 2^N \times \frac{KSp(n-1)-Q2}{2^{N-1}-Q2} & if\ Q2 < KSp(n-1) \le 2^{N-1} \\ 2^N \times \frac{2^N - KSp(n-1)-Q2}{2^{N-1}-Q2} & if\ 2^{N-1} < KSp(n-1) \le 2^N - Q2 \\ 2^N \times \frac{2^N - KSp(n-1)}{Q2} & if\ 2^N - Q2 < KSp(n-1) \le 2^N - 1 \\ 2^N - 1 - Q2 & otherwise \end{cases} \quad (5)$$

where $Q2$ represents the control parameter of the *DPWLCmap*. *KSp(n)* and *KSp*$(n-1)$ are the outputs of *DPWLCmap* at the nth and $(n-1)$th iterations, respectively. $Q2$, *KSp(n)*, and *KSp*$(n-1)$, and range between 1 to 2^{N-1}.

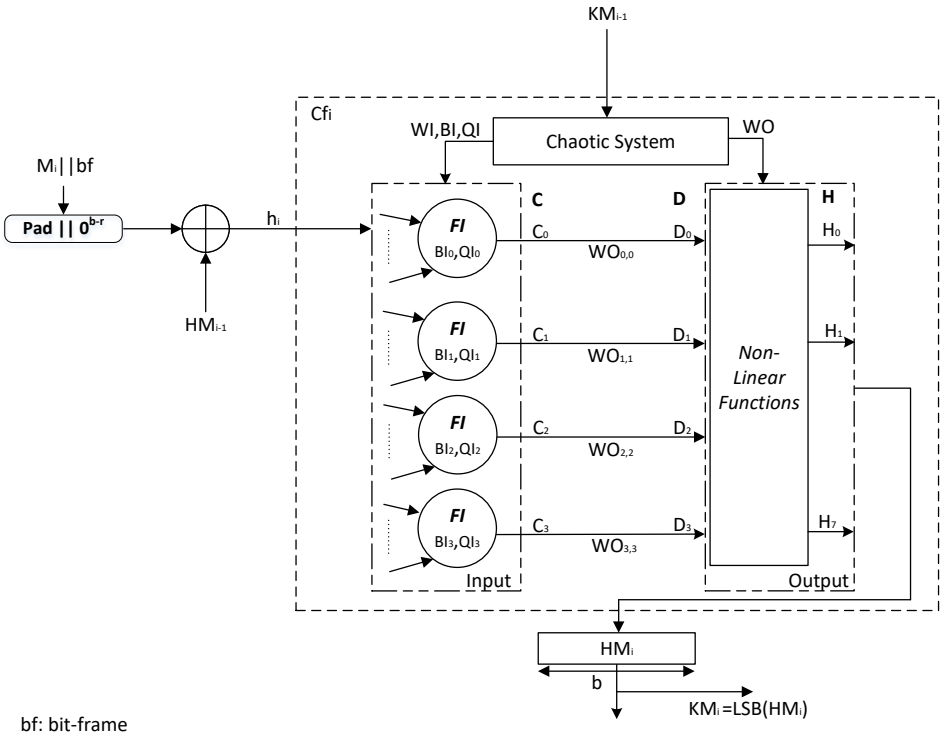


Figure 4. Detailed structure of the ith *Chaotic compression function* of the proposed *CNN-Duplex* based on a one-layered *NL* for *AEADS*.

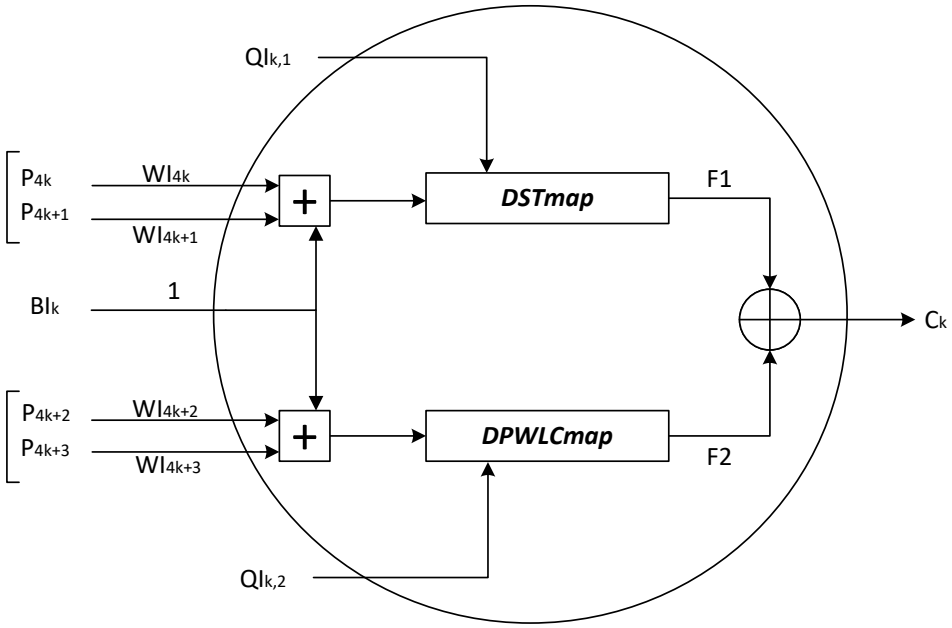


Figure 5. Detailed structure of the *k*th input neuron for the first choice of the proposed *CNN-Duplex*.

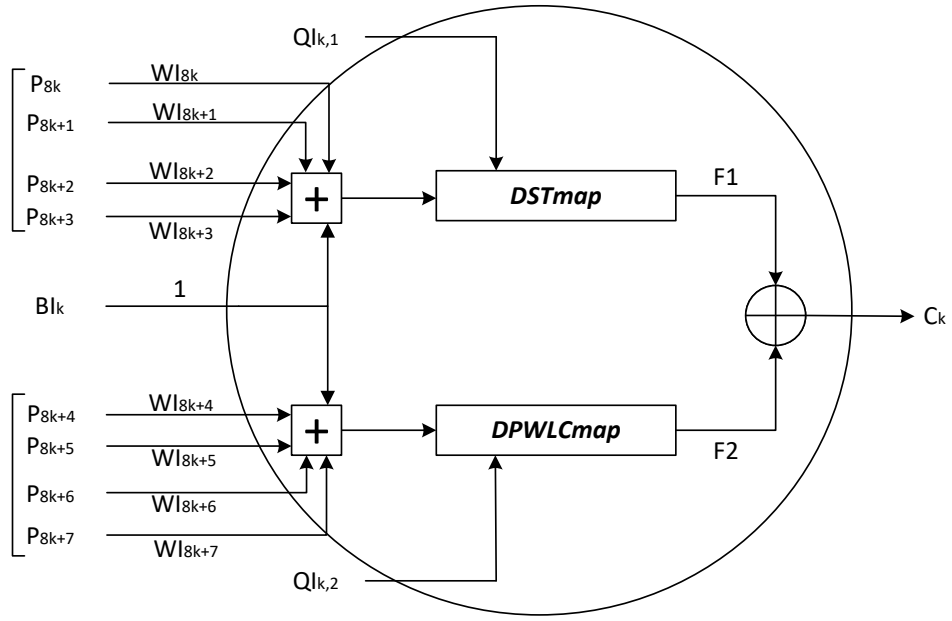


Figure 6. Detailed structure of the *k*th input neuron for the second choice of the proposed *CNN-Duplex*.

After computing, both outputs of the chosen chaotic maps *DSTmap* and *DPWLCmap* are combined together using XOR operation to calculate the outputs of neurons denoted by the parameter C_k, $(k = 0, \ldots, 3)$, which is given by Equation (6) for the first choice, and by Equation (7) for the second choice.

$$\begin{aligned} & C_k = mod\{[Fn1 + Fn2], 2^N\} \\ & where \begin{cases} Fn1 = DSTmap\{BI_k + mod([\sum_{j=4k}^{4k+1}(P_j \times WI_j)], 2^N), QI_{k,1}\} \\ Fn2 = DPWLCmap\{BI_k + mod([\sum_{j=4k+2}^{4k+3}(P_j \times WI_j)], 2^N), QI_{k,2}\} \end{cases} \end{aligned} \tag{6}$$

$$C_k = mod\{[Fn'1 + Fn'2], 2^N\}$$
$$where \begin{cases} Fn'1 = DSTmap\{BI_k + mod([\sum_{j=8k}^{8k+3}(P_j \times WI_j)], 2^N), QI_{k,1}\} \\ Fn'2 = DPWLCmap\{BI_k + mod([\sum_{j=8k+4}^{8k+7}(P_j \times WI_j)], 2^N), QI_{k,2}\} \end{cases} \quad (7)$$

The values of $C_k, (k = 0, \ldots, 3)$, weighted by $WO_{k,k}, (k = 0, \ldots, 3)$, form the inputs $D_k, (k = 0, \ldots, 3)$ of the *NL* output layer: $D_k = C_k \times WO_{k,k}, (k = 0, \ldots, 3)$, which are truncated to 32-bit lengths. The final output values $H_k, (k = 0, \ldots, 7)$ are given by the following equation (see Figure 7):

$$\begin{cases} H_0 = t2 \oplus t1 \oplus D_0 \\ H_1 = t1 \oplus D_0 \\ H_2 = D_1 \oplus D_0 \\ H_3 = D_2 \oplus D_1 \\ H_4 = D_3 \oplus D_2 \\ H_5 = t1 \oplus D_1 \oplus D_0 \\ H_6 = t1 \oplus D_2 \oplus D_1 \\ H_7 = t1 \oplus D_3 \oplus D_2 \end{cases} \quad where \begin{cases} t1 = \Sigma1(D_3) \oplus Ch(D_1, D_2, D_3) \\ t2 = \Sigma0(D_1) \oplus Maj(D_1, D_2, D_3) \end{cases} \quad (8)$$

The $H_k, (k = 0, \ldots, 7)$ are values of 32-bit size.

The four non-linear functions ($\Sigma0$ and $\Sigma1$, *Ma*, and *Ch*) are represented as follows:

$$\begin{cases} \Sigma0(D_1) = ROTR^2(D_1) \oplus ROTR^{13}(D_1) \oplus ROTR^{22}(D_1) \\ \Sigma1(D_3) = ROTR^6(D_3) \oplus ROTR^{11}(D_3) \oplus ROTR^{25}(D_3) \\ Ma(D_1, D_2, D_3) = (D_1 \wedge D_2) \oplus (D_1 \wedge D_3) \oplus (D_2 \wedge D_3) \\ Ch(D_1, D_2, D_3) = (D_1 \wedge D_2) \oplus (\neg D_1 \wedge D_3) \\ ROTR^n(x) = (x \gg n) \vee (x \ll (32 - n)) \end{cases} \quad (9)$$

where $\wedge$: *AND logic*, $\neg$: *NOT logic*, $\oplus$: *XOR logic*, $\vee$: *OR logic*, $\gg$: *Binary Shift Right operation and* $\ll$: *Binary Shift Left operation*.

In the proposed *AEADS*, and in order to calculate the *b*-bit intermediate hash values, we first iterate the output layer n_r times, with n_r = 1, 8, 24, according to the required security level and speed. Then, with the chosen value of n_r, we iterate again the output layer twice for the first choice (r = 256 bits, c = 256 bits) and four times for the second choice (r = 512 bits, c = 512 bits) to obtain the intended *b*-bit intermediate hash values.

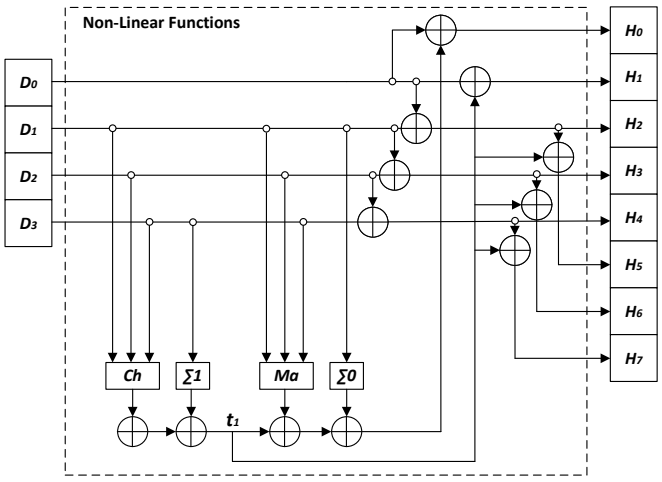


Figure 7. Detailed structure of *NL* Functions block.

4. Performance Analysis of the Proposed *AEADS*

In this section, we will first assess the security of the proposed *AEADS* against known cryptanalytic attacks. After that, we calculate some statistical tests [32].

4.1. Cryptanalytic Analysis

This section examines the security of all components of the proposed *AEADS*. Indeed, we calculate the key space and we test the key sensitivity, the message sensitivity, the collision resistance, the diffusion effect in order to confirm the resilience of the proposed *AEADS* against most cryptanalytic attacks.

4.1.1. Key Space

The length of secret key K, composed by all parameters of the system and by all initial conditions, is equal to 160 bits. It means that the brute force attack of the proposed *AEADS* is impracticable.

4.1.2. Key Security and Sensitivity

An *AEADS* must be very sensitive to one bit change on the used key K. This property is important in order to test the resistance of the proposed algorithm against many different attacks [33]. Consequently, it is necessary to test the sensitivity of generated ciphertexts and obtained hash values when a slight change in the secret K occurred. Indeed, it is impossible from the generated sequences to calculate the secret key K due to the structure of the chaotic generator [34]. Therefore, to check the key sensitivity of ciphertexts, we calculate the following three parameters: Number of Pixel Change Rate (*NPCR*), Unified Average Changing Intensity (*UACI*), and Hamming Distance (*HD*) . The two parameters *NPCR* and *UACI* are introduced by Biham et al. [35] and are given by the following equations:

$$NPCR = \frac{1}{P \times L \times C} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} D(i,j,p) \times 100\% \tag{10}$$

where

$$D(i,j,p) = \begin{cases} 0 & if\ C_1(i,j,p) = C_2(i,j,p) \\ 1 & if\ C_1(i,j,p) \neq C_2(i,j,p) \end{cases} \tag{11}$$

$$UACI = \frac{1}{P \times L \times C \times 255} \times \sum_{p=1}^{P} \sum_{i=1}^{L} \sum_{j=1}^{C} |C_1(i,j,p) - C_2(i,j,p)| \times 100\% \tag{12}$$

In Equations (10)–(12), the three variables i, j, and p represent the row, column, and plane indexes of the image, respectively. Additionally, the three values L, C, and P are, the length, width, and plane sizes of the image, respectively. Indeed, the two optimal values of *NPCR* and *UACI* are equal to 99.61% and 33.46%, respectively [36].

The parameters (*NPCR*, *UACI*) are necessary to ensure that the proposed *AEADS* is resistant against the key sensitivity attack but insufficient condition. For this reason, we calculate the Hamming Distance (*HD*) measure [37]. This measure is represented by the following equation:

$$HD(C_1, C_2) = \frac{1}{|Ib|} \times \sum_{K=1}^{|Ib|} (C_1(K) \oplus C_2(K)) \tag{13}$$

where $|Ib| = L \times C \times P \times 8$ (in bits) is the length of the image.

The optimal *HD* value is equal to 50%. So, a good *AEADS* must produce an *HD* value very close to 50% [38].

Table 1 shows that the values of three parameters *NPCR*, *UACI*, and *HD* of the proposed *AEADS* are very close to the optimal values. As a result, a very high resistance to different attacks is achieved.

Table 1. The *NPCR*, *UACI* and *HD* of the proposed *AEADS*.

AEAD	*NPCR*	*UACI*	*HD* %
Proposed *AEADS*	99.67	33.43	49.8

On the other hand, to check the key sensitivity of the authentication tag T with respect to the secret key K, we compute the authentication tags T_i (in hexadecimal format), the number of changed bits $B_i(T, T_i)$ (bits), and the Hamming Distance $HD_i(T, T_i)(\%)$, for the following input message M "With the wide application of Internet and computer technique, information security becomes more and more important. As we know, hash function is one of the cores of cryptography and plays an important role in information security. Hash function takes a message as input and produces an output referred to as a hash value. A hash value serves as a compact representative image (sometimes called digital fingerprint) of input string and can be used for data integrity in conjunction with digital signature schemes." and under each of the five following conditions:

Condition 1: We use the original secret key K.

In each of the following conditions, we flip the Least Significant Bit (LSB) in the above-mentioned initial conditions and parameters:

Condition 2: We change the value of initial condition $KSs(0)$ in the secret key K.
Condition 3: We change the value of parameter Ks in the secret key K.
Condition 4: We change the value of initial condition $KSs(-1)$ in the secret key K.
Condition 5: We change the value of control parameter $Q1$ in the secret key K.

We represent the results obtained, under each condition, of T_i, B_i, and $HD_i(\%)$ for the two width values (b = 64 bytes and 128 bytes) with 256-bit authentication tag length in Table 2. For the two width values, all results are very close to the expected theoretical values (B_i = 128 bits, and HD_i = 50%), demonstrating the high key sensitivity of the proposed *AEADS*.

Table 2. The key sensitivity of 256-bit authenticated tag T to K for the two width values (64 bytes and 128 bytes).

	Message Variants	**Authenticated Tag in Hexadecimal Format**	B_i	HD_i%
b = 64 bytes	1	5cf839f23ab09f77a0f5efb4b8376f7014a64d4573a0a49d6b622459c7f3066c	–	–
	2	2aa99d0a65ba78b1dca34c4737dab62f10da532481ac655b6ad59f334dd57a38	133.00	51.95
	3	635f0c9970b9cee82508fc620e4481b1db50e8250ac1b5e1218dc832f499a1a6	141.00	55.07
	4	1dad759ec9f258d6022b272a97ffb69d70c543d49d47591f0893c8a8efa93de1	133.00	51.95
	5	9b498e512ec5e2e37ea0c0791124170422488f0f2b7c13c044f8c4bd77945cb6	138.00	53.90
	Average	–	136.25	53.22
b = 128 bytes	1	c1fb8921581959928cce8d75b42e6c5d4dff037b91e42ca4a904cd8c50d5a2aa	–	–
	2	7fc5609746e964392d0ff999124274d627c8bb57de6ef906e1237449d4a9bbc8	128.00	50.00
	3	404864221fbfdd28a5cc0af04543a8d9634dbc41d62abf9b36a51db210ab092a	124.00	48.44
	4	d7366dee282cfa69a48c55e381255d3e79505c83a5f6a224805faaf9782aab44	128.00	50.00
	5	58d7387465730f264dbb1161ffdcc4d6ba31ab63f2ee6e1a4089bbad0564d418	126.00	49.21
	Average	–	126.50	49.41

4.1.3. Message Sensitivity

An *AEADS* should be very sensitive to the input message M. It means that a little change in its input message would generate a completely different ciphertext and Tag. To check this property, we measure the ciphertext and the authenticated Tag (in hexadecimal format), the number of changed bits $B_i(T, T_i)$ (in bits), and the sensitivity of the ciphertext and the authenticated tag to the initial message M (in %) for a given secret key K, by using the Hamming Distance $HD_i(T, T_i)$ (in %) metric as follows:

$$B_i(T, T_i) = \sum_{k=1}^{|T|} [T(k) \oplus T_i(k)] \ (bits) \tag{14}$$

and

$$HD_i(T, T_i)_{\%} = \frac{B_i(T, T_i)}{|T|} \times 100\% \tag{15}$$

Under the following conditions, we obtain the different message variants used for testing:

Condition 1: The initial message M is the one used in Section 4.1.2.
Condition 2: We change the first character `"W"` to `"X"` in the input message.
Condition 3: We change the word `"With"` to `"Without"` in the input message.
Condition 4: We change the dot (".") to comma (",") at the end of the input message.
Condition 5: We add a `"blank space"` at the end of the input message.
Condition 6: We swap the first block M_1 of the input message

"With the wide application of Internet and computer technique, information security becomes more and more important. As we know, hash function is one of the cores of cryptography and plays an important role in information security. Hash function takes a mes,"

With the second block M_2 of the same input message

"sage as input and produces an output referred to as a hash value. A hash value serves as a compact representative image (sometimes called digital fingerprint) of input string and can be used for data integrity in conjunction with digital signature schemes."

Under each condition, we show the results of T_i (in hexadecimal format), B_i (in bits), and HD_i (in percentage) obtained for the two width values (b = 64 bytes and 128 bytes) with T = 256 bits in Table 3. For the two width values, all results are very close to the expected theoretical values (B_i = 128 bits, and HD_i = 50%), demonstrating the high message sensitivity of the proposed *AEADS*.

Table 3. The message sensitivity of 256-bit authenticated tag T to M for the two width values (64 bytes and 128 bytes).

	Message Variants	Authenticated Tag in Hexadecimal Format	B_i	HD_i%
b = 64 bytes	1	5cf839f23ab09f77a0f5efb4b8376f7014a64d4573a0a49d6b622459c7f3066c	–	–
	2	af1f98eda7a69b2a1c9b146d60b04e7f3a1c421ead01e913d12ab3018b86bf60	135.00	52.73
	3	ac3a80e98af6809e94bd50b7c337acfa6987fce4df7534b388ffc8fe224f98e0	127.00	49.60
	4	20aaf22ec8f025fbb8da59b6f175dc5587b1e19c60c0603c296fe5df5fabd816	114.00	44.53
	5	a177b89d7ec17468ff0c4aff348ba20334855f53f5c9bcdb3e4e54273e40a977	133.00	51.95
	6	0bdc884523ac64cc5c14150af8068b1b46f83f0067c7b3d7c3d52dc67d2b99dd	136.00	53.13
	Average	–	129.00	50.39
b = 128 bytes	1	c1fb8921581959928cce8d75b42e6c5d4dff037b91e42ca4a904cd8c50d5a2aa	–	–
	2	55d53d2a5f8b2ce215f0ecd097296d26c1511f8f49c21a82cb1b9b749d63e9dd	124.00	48.43
	3	7e428a69f21099ce7717ae2892e8107160a8ecd18e757d856b8ac3dc99ca3f7c	127.00	49.60
	4	e306a1484f36d072c37c9f33c8e18cbb987710db2ecc56862551450e75c7d96e	116.00	45.31
	5	2622493df19883b33d23aa780a3fe817f1aa1c79c869ddc8ff759fa704e06bc9	121.00	47.26
	6	504ab6f4aa4b2262727bf6a067b83f9679aeada6f2046386e7c7aab0f9d13880	137.00	53.51
	Average	–	125.00	48.82

4.1.4. Collision Resistance Analysis

In the cryptography field, collision resistance is a very important property of cryptographic hash functions. This statistical analysis evaluates the collision resistance property quantitatively [39]. To do this test, we calculate the number of hits ω as defined in Equation (16), given the authentication tag $T = \{c_1, c_2, \ldots, c_s\}$ of a random input message M (in the ASCII format), and its corresponding authentication tag $T' = \{c'_1, c'_2, \ldots, c'_s\}$ generated after flipping only one bit of the same input message M.

$$\omega = \sum_{i=1}^{s} f(D(c_i), D(c_i^{'})), \tag{16}$$

where the function f is represented as follow:

$$f(x,y) = \begin{cases} 0 & \text{if } x \neq y; \\ 1 & \text{if } x = y. \end{cases} \tag{17}$$

Note that, the value of s is equal to $\frac{u}{8}$, and the function $D(.)$ converts the hexadecimal inputs to their equivalent decimal values. In theory, the relation between a number of hits $\omega = 0, 1, 2, \ldots, s$ and a number of independent experiments J, as mentioned in [40], is represented by the following equation:

$$W_J(\omega) = J \times Prob\{\omega\} = J \frac{s!}{\omega!(s-\omega)!} (\frac{1}{2^k})^{\omega} (1 - \frac{1}{2^k})^{s-\omega}, \tag{18}$$

In fact, the theoretical values of $W_J(\omega)$, calculated according to Equation (18), are represented in Table 4.

Table 4. Theoretical values of the number of hits ω for T = 256 bits with respect to the number of tests J.

		ω				
		0	**1**	**2**	**3**	**32**
J	2048	1806.91	226.74	13.78	0.54	1.76×10^{-74}
	1024	903.45	113.37	6.89	0.27	8.84×10^{-75}
	512	451.72	56.68	3.44	0.13	4.42×10^{-75}

For the two choices of width b, the results obtained are given in Table 5. In fact, we obtain comparable results as expected. Then, the absolute difference d of the two authentication tags is defined as follow:

$$d = \sum_{i=1}^{s} |D(c_i) - D(c_i^{'})|. \tag{19}$$

The minimum, maximum, mean, and mean/character of d for the two lengths of b with J = 2048 tests are presented in Table 6. As observed from the obtained results, it is clear that the values of mean/character are very close to the expected ones that is equal to 85.33 for 256-bit authentication tag size (L = 256) [41]. Indeed, the expected value is evaluated by the following equation:

$$E[D(c_i) - D(c_i^{'})] = \frac{1}{3} \times L \tag{20}$$

Table 5. Number of hits ω for J = 2048 tests with respect to the width length b.

	ω					
	0	**1**	**2**	**3**	**4**	**5**
b = 64 bytes	1816	218	13	1	0	0
b = 128 bytes	1898	142	8	0	0	0

Table 6. Minimum, maximum, mean, and mean/character of d for the two lengths of b with J = 2048 tests.

	***b* (bytes)**	**Minimum**	**Maximum**	**Mean**	**Mean/Character**
Proposed *AEADS*	64	1642	3784	2665.24	83.28
	128	1846	3548	2668.15	83.33

4.1.5. The Diffusion Effect

The optimal value of the diffusion effect (mentioned also as the *Strict Avalanche Criterion* (*SAC*) in literature [42]) is obtained when swapping any bit in the original input message M that causes a variation in each output bit in the authenticated tag T with a probability equal to 50% [43]. To check the performance of the proposed *AEADS*, the next diffusion test is executed. First, the authentication tag T for the previous input message M is produced. Then, a new authentication tag T' for the same input message M with one changed bit at random is generated. Next, the B_i between the two obtained authentication tags T and T' is computed. This test is repeated J times, with J equal to 512, 1024, and 2048 tests. At the end, we calculate the six statistical tests represented below:

1. $\bar{B} = \frac{1}{J}\sum_{i=1}^{J} B_i$ bits: Mean number of bits changed;
2. $P = (\frac{\bar{B}}{u}) \times 100\%$: Mean changed probability (mean of $HD_i(\%)$);
3. $B_{min} = min(\{B_i\}_{i=1,\ldots,J})$ bits: Minimum number of bits changed;
4. $B_{max} = max(\{B_i\}_{i=1,\ldots,J})$ bits: Maximum number of bits changed;
5. $\Delta B = \sqrt{\frac{1}{J-1}\sum_{i=1}^{J}(B_i - \bar{B})^2}$: Standard variance of the changed bit number;
6. $\Delta P = \sqrt{\frac{1}{J-1}\sum_{i=1}^{J}(\frac{B_i}{u} - P)^2} \times 100\%$: Standard variance of the changed probability.

The results presented in Table 7 with J equal to 2048 tests demonstrate that the diffusion effect values, for the 256-bit authentication tag length, are very close to the expected results ($\bar{B}$ = 128 bits, and P = 50% for the proposed *AEADS*). Additionally, it is noted that both $\bar{B}$ and P are very close to the ideal values, while ΔB and ΔP are very small, which means that the diffusion effect is highly stable.

Table 7. Statistical test of the diffusion effect results for proposed *AEADS* with J = 2048 tests.

	b (bytes)	$\bar{B}$	P	B_{min}	B_{max}	ΔB	ΔP
Proposed *AEADS*	64	128.10	50.04	101	155	7.96	3.13
	128	128.01	50.00	104	150	7.8	2.23

4.2. Statistical Tests

As it is difficult to provide mathematical proof for cryptographic algorithms, the proposed *AEADS* is evaluated mainly using statistical tests. Consequently, we applied *NIST* test, histogram, chi-square, entropy, and correlation analysis.

4.2.1. NIST Test

Indeed, the *NIST* statistical test is used to evaluate the performance of the ciphertext generated, and is also considered one of the most popular standards that is used for investigating the randomness of binary data [44]. The *NIST* test is applied to many ciphertexts, and all the *NIST* values obtained are good *NIST* results (as expected), which means that the ciphertexts have a very high randomness. In fact, we present in Figure 8 one of the results obtained after applying the *NIST* statistical package.

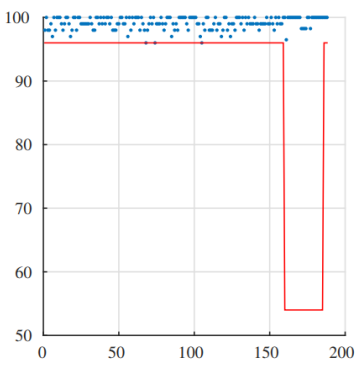


Figure 8. *NIST* test for one of ciphertext results.

4.2.2. Histogram and Chi-Square Analysis

An *AEAD* algorithm is considered to be very strong against different statistical attacks, if the histogram of the ciphertext is uniformly distributed. In fact, the uniformity test is essential for vision, but it is not sufficient. For this reason, we apply the chi-square test (defined by the Equation (21)) in order to confirm the uniformity of the histogram statistically:

$$\chi^2_{exp} = \sum_{i=0}^{Nb-1} \frac{(o_i - e_i)^2}{e_i} \tag{21}$$

where the number of levels Nb is equal to 256 in this case, o_i is the observed occurrence frequency of each color level (from 0 to 255) on the histogram of the encrypted image, and e_i is the expected occurrence frequency of the uniform distribution, given in this case by the following equation:

$$e_i = \frac{L \times C \times P}{Q} \tag{22}$$

For a secure *AEADS*, the theoretical chi-square value, which is equal to 293 in the case of $\alpha = 0.05$ and $Nb = 256$, must be more than the experimental chi-square one. In fact, the histograms os the plain/cipher images for Lena, Boat and Camera man (C-man) images, having a length equal to 512x512x3, are given in Figures 9–11. As we can see from the figures, the histograms of the ciphered images appear to be uniform. Additionally, in order to verify the uniformity, the chi-square test with number of classes equal to 256 and $\alpha = 0.05$, is performed (see Table 8). As a result, the theoretical value are larger than the experimental one, which means that the histograms have a uniform distribution.

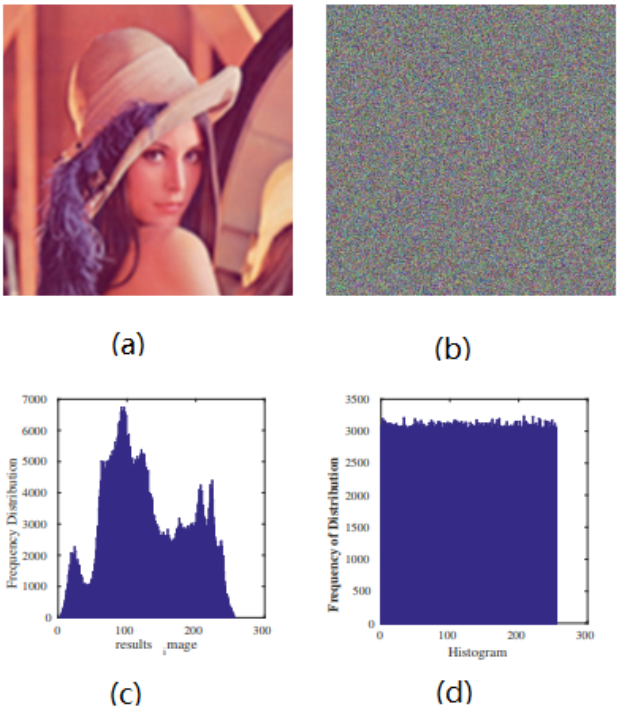


Figure 9. Results of Lena image. (**a**) Lena image, (**b**) Ciphered Lena, (**c**) Histogram of Lena image, and (**d**) Histogram of ciphered Lena.

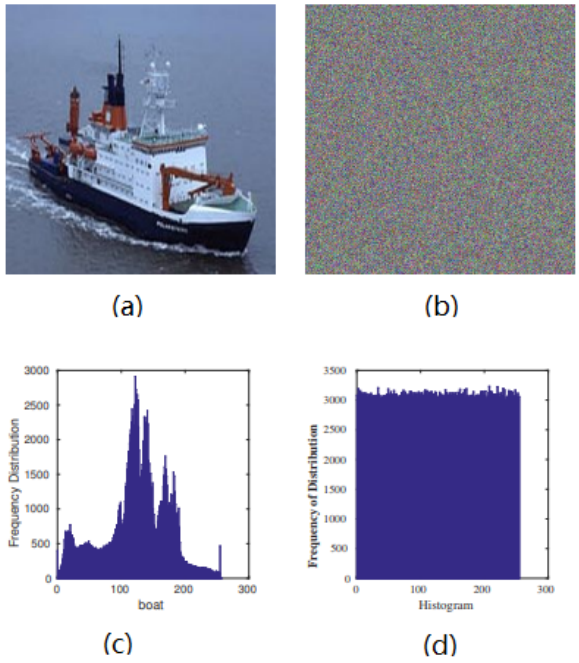


Figure 10. Results of Boat image. (**a**) Boat image, (**b**) Ciphered Boat, (**c**) Histogram of Boat image, and (**d**) Histogram of ciphered Boat.

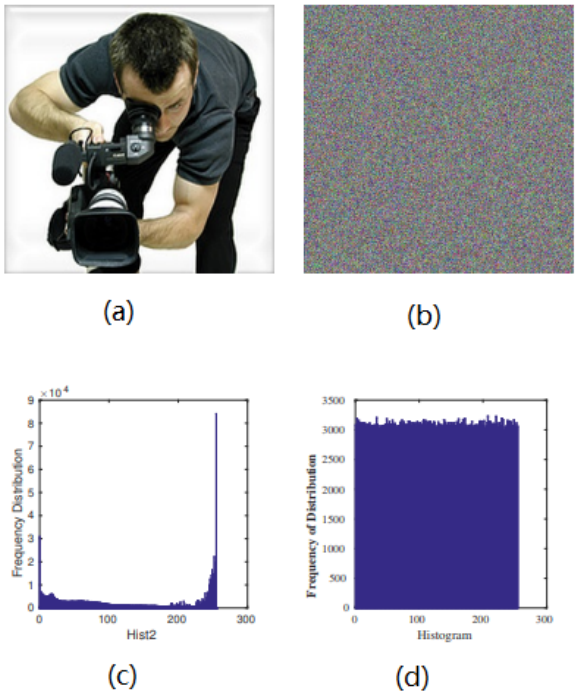


Figure 11. Results of Camera man image. (**a**) Camera man image, (**b**) Ciphered Camera man, (**c**) Histogram of Camera man image, and (**d**) Histogram of ciphered Camera man.

Table 8. The chi-square results for the ciphered images: Lena, Boat, and Camera man.

Images	Experimental Values	Theoretical Values
Lena 512 × 512 × 3	272.152689	293.247835
Boat 512 × 512 × 3	268.465369	293.247835
C-man 512 × 512 × 3	270.317852	293.247835

4.2.3. Entropy Analysis

The random behavior of the encrypted image can be quantitatively measured by the entropy information given by Shannon [45]:

$$H(C) = -\sum_{i=0}^{N_c-1} P(c_i) \times \log_2(P(c_i)) \tag{23}$$

where $H(C)$ is the entropy value of the ciphered image, and $P(c_i)$ ($c_i = 0, 1, \ldots, 255$) is the probability of each gray level appearance. Note that, the entropy has maximum value equal to 8 in the case of equal probability levels. On the other hand, the encryption algorithm is more robust when the experimental entropy value is closer to the maximum value. In Table 9, we give the results received from the entropy test applied on the plain and ciphered images. From this table, it is clear that the obtained entropy values of encrypted images are very close to the optimal value, which means that the proposed *AEADS* has high levels of resilience.

Table 9. Entropy results for the ciphered images: Lena, Boat, and Camera man.

Entropy	Lena	Boat	Camera Man
Plain image	7.4504	7.6845	7.4892
Cipher image	7.9584	7.9865	7.9582

4.2.4. Correlation Analysis

The correlation analysis is one of the statistical tests that are used to test the immunity of the cryptosystem against cryptanalysis. The attacker must not have any partial information on the original plain image or any information of the used secret key. This means that, the ciphered image should be extremely different from its original version. This property can be measured by correlation analysis. Indeed, it is well-known that neighboring pixels are very correlated and redundant in the plain images. Thus, the correlation and the redundancy of the neighboring pixels should be as low as possible in the ciphered images. In order to calculate the correlation coefficient, we use the following four equations [46]:

$$\rho_{xy} = \frac{cov(x,y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{24}$$

where

$$cov(x,y) = \frac{1}{N}\sum_{i=1}^{N}([x_i - E(x)][y_i - E(y)]) \tag{25}$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \tag{26}$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{27}$$

In the equations before, x_i and y_i are the values of the two neighboring pixels in the encrypted image or the corresponding plain image. To test the security of our proposed *AEADS* concerning this kind of attack, we first set the value of N to 10,000 pairs of neighboring pixels in horizontal, vertical, and diagonal directions from the plain image and its encrypted version. We show in Table 10 the correlation coefficients of two neighboring pixels in the plain and encrypted images. As we can see from this table, these results are very close to the optimal values.

Table 10. Entropy results obtained.

Image	Horizontal	Vertical	Diagonal
Lena	0.939403	0.971060	0.931085
Lena encrypted	−0.002674	−0.009113	−0.002178
Boat	0.951837	0.962357	0.915555
Boat encrypted	−0.002602	−0.009819	0.002402
Camera man	0.887794	0.734230	0.888141
Camera man encrypted	−0.007750	−0.006998	−0.002788

5. Performance Comparison with Other *AE* Algorithms

This section presents the different characteristics of essential *AEAD* algorithms in the literature. Additionally, it provides a comparison for our proposed *AEADS* from a security aspect with the main three modes such as; *GCM* mode, *CCM* mode, and *OCB* mode.

5.1. Characteristic Comparison

Table 11 summarizes the characteristics of the essential *AEAD* algorithms. In fact, the characteristics mentioned in this section are taken from the basic list of properties required for *NIST* proposal and extended by features, which varies across modes [47,48].

Table 11. Characteristic comparison between our proposed *AEADS* and other *AEAD* schemes.

Feature	Proposed *AEADS*	GCM	CCM	OCB
Tag length (bits)	256	0 to n	$16k, k \in \{2, \ldots, 8\}$	0 to n
IV size (bits)	Any	Any (favored: $n - 32$)	n	n
Number of passes	One	Two	Two	One
Provably secure	yes	yes	yes	yes
Keying material	one key	one key	one key	one key
Online	no	yes	no	yes
Endian dependent	yes	yes	yes	no
Incremental MAC	yes	yes	no	no
Error propagation	no	no	no	no
Associated data authentication	yes	yes	yes	no

5.2. Security Comparison

Table 12 presents the security comparison of the three essential *AEAD* modes with our proposed *AEADS*. It shows that our proposed *AEADS*, similarly to other three-modes, is secure against the chosen plaintext attack.

Table 12. Performance comparison between our proposed *AEADS* and other *AEAD* schemes.

Feature	Proposed *AEADS*	GCM	CCM	OCB
Secure against chosen-plaintext attack	yes	yes	yes	yes
Synchronization between sender and receiver	Same *IV*	Same *IV*	Same nonce	Same nonce
Keying Requirements	One block cipher key	One block cipher key	One block cipher key	One block cipher key
Message Length Requirements	Any bit string allowed	Arbitrary message up to 2^{39}–256 bits Arbitrary authenticated data up to 2^{64} bits	Arbitrary message up to 2^{8L} bits where $L = 2, \ldots, 8$ Arbitrary authenticated data up to 2^{64} bits	Any bit string allowed
Underlying Cipher Block Size Requirements (bits)	512, 1024	64, 128	Only 128	128, 192, 256
Parallelizability	None	Encryption block level Authentication bit level	None	Fully parallelizable

Additionally, these mentioned modes use one block cipher key mechanism that requests for the memory resources to save the key. In addition to the *CCM* mode, the other three modes are parallelizable. Our proposed scheme can be also implemented in a parallel manner, which will be useful for scenarios having powerful nodes in the network, such as a multi-core base station processing many large packets resulting from data aggregation [49]. The authenticity of the ciphertext depends on the *IV* or the nonce in the three modes, while it depends on the *IV* and *K* for our proposed *AEADS*.

Therefore, these values must be kept identical in encryption and decryption processes. For Message Length Requirements, our proposed *AEADS*, like the *OCB* mode, allows any bit string, it can be satisfied by different applications. The *CCM* mode requests the message length must be multiples of 16, it is certainly to restrict the range of application.

6. Conclusions and Future Work

In this paper, we have designed, realized, and evaluated the robustness of a new Authenticated Encryption with Associated Data Scheme (*AEADS*) using the Duplex construction based on Chaotic Neural Networks provided by a chaotic system. The proposed scheme is composed of both encryption and decryption processes. In the encryption process the inputs are composed of *IV*, *K*, *AD*, and *M*, while the outputs are *C* and *T*. These outputs, in addition to *IV*, *K*, and *AD*, form the inputs of the decryption process. Then, the initial message *M* is decrypted if the calculated tag T' is equal to the received tag *T*. Otherwise, an error message is delivered. These two processes are realized for the two-width length 64 bytes and 128 bytes. On the other hand, we tested our proposed scheme against the different cryptanalytic attacks. In fact, we evaluated the key and the message sensitivity, the collision resistance analysis and the diffusion effect. Additionally, we tested *AEADS* using the different statistical tests such as *NIST*, histogram, chi-square, entropy and correlation analysis. All the results obtained demonstrate that the security of the proposed system, against a battery of cryptanalytic and statistic attacks, has been achieved. We conclude that the proposed scheme simultaneously ensures the confidentiality and integrity of data transmitted over unsecured channels and the authenticity of the data source. Our future work will focus on an FPGA-based implementation of the system, the evaluation of its hardware performance and the comparison of our *AEADS* with other authenticated encryption modes.

Author Contributions: Conceptualization, N.A., S.E.A., T.M.H., O.D., R.A. and M.K.; methodology, N.A. and S.E.A.; software, N.A.; validation, N.A., S.E.A. and T.M.H.; formal analysis, N.A., S.E.A., T.M.H., O.D. and R.A.; writing—original draft preparation, N.A. and S.E.A.; writing—review and editing, N.A., S.E.A. and T.M.H.; visualization, N.A.; supervision, S.E.A., T.M.H., O.D., R.A. and M.K.; project administration, S.E.A. and T.M.H.; funding acquisition, T.M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Hanoi University of Science and Technology (HUST) under project number T2020-SAHEP-008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Stallings, W. *Cryptography and Network Security: Principles and Practice*; Pearson: Upper Saddle River, NJ, USA, 2017.
2. Abdoun, N.; El Assad, S.; Taha, M.A.; Assaf, R.; Deforges, O.; Khalil, M. Hash Function based on Efficient Chaotic Neural Network. In Proceedings of the 2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, UK, 14–16 December 2015; pp. 32–37.
3. Abdoun, N.; El Assad, S.; Taha, M.A.; Assaf, R.; Déforges, O.; Khalil, M. Secure hash algorithm based on efficient chaotic neural network. In Proceedings of the 2016 International Conference on Communications (COMM), Bucharest, Romania, 9–10 June 2016.

4. McGrew, D.; Paterson, K. Authenticated Encryption with AES-CBC and HMAC-SHA. Internet Engineering Task Force (IETF)–draft-mcgrew-aead-aes-cbc-hmac-sha2-01. 2012. Available online: https://datatracker.ietf.org/doc/html/draft-mcgrew-aead-aes-cbc-hmac-sha2-01 (accessed on 10 November 2021).
5. Rajashree, S.; Sukumar, R. CBC (Cipher Block Chaining)-Based Authenticated Encryption for Securing Sensor Data in Smart Home. In *Smart IoT for Research and Industry*; Springer: Cham, Switzerland, 2022; pp. 189–204.
6. Kavun, E.B.; Mihajloska, H.; Yalçin, T. A Survey on Authenticated Encryption–ASIC Designer's Perspective. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–21. [CrossRef]
7. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Proceedings of the International Workshop on Selected Areas in Cryptography, Toronto, ON, Canada, 11–12 August 2011; pp. 320–337.
8. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Ascon v1.2. Submission to the CAESAR Competition. Available online: https://competitions.cr.yp.to/round3/asconv12.pdf (accessed on 9 November 2021).
9. Bao, Z.; Chakraborti, A.; Datta, N.; Guo, J.; Nandi, M.; Peyrin, T.; Yasuda, K. PHOTON-beetle authenticated encryption and hash family. *NIST Lightweight Compet. Round* **2019**, *1*, 115.
10. Bhattacharjee, A.; List, E.; Lpez, C.; Nandi, M. *The Oribatida Family of Lightweight Authenticated Encryption Schemes*; Indian Statistical Institute Kolkata: Kolkata, India; p. 2019.
11. Khan, S.; Lee, W.K.; Hwang, S.O. Scalable and Efficient Hardware Architectures for Authenticated Encryption in IoT Applications. *IEEE Internet Things J.* **2021**, *8*, 11260–11275. [CrossRef]
12. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Cryptographic Sponges. 2011. Available online: http://sponge.noekeon.org (accessed on 10 August 2021).
13. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Permutation-based encryption, authentication and authenticated encryption. *Dir. Authenticated Ciphers* **2012**, 159–170.
14. Borowski, M. Cryptographic Applications of the Duplex Construction. *Ann. Univ. Mariae Curie-Sklodowska Sect. AI* **2014**, *14*, 37–48. [CrossRef]
15. Chang, D. Sufficient Conditions on Padding Schemes of Sponge Construction and Sponge-Based Authenticated-Encryption Scheme. In Proceedings of the International Conference on Cryptology in India, Kolkata, India, 9–12 December 2012; pp. 545–563.
16. Morawiecki, P.; Gaj, K.; Homsirikamol, E.; Matusiewicz, K.; Pieprzyk, J.; Rogawski, M.; Srebrny, M.; Wójcik, M. ICEPOLE: High-speed, hardware-oriented authenticated encryption. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Busan, Korea, 23–26 September 2014; pp. 392–413.
17. Abdoun, N.; El Assad, S.; Hammoud, K.; Assaf, R.; Khalil, M.; Deforges, O. New keyed chaotic neural network hash function based on sponge construction. In Proceedings of the 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), Cambridge, UK, 11–14 December 2017; pp. 35–38.
18. Abdoun, N.; El Assad, S.; Assaf, R.; Déforges, O.; Khalil, M.; Belghith, S. *Design and Implementation of Robust Keyed Hash Functions Based on Chaotic Neural Network*; Electronics; Universite de Nantes: Nantes, France, 2018.
19. De la Fraga, L.G.; Mancillas-López, C.; Tlelo-Cuautle, E. Designing an authenticated Hash function with a 2D chaotic map. *Nonlinear Dyn.* **2021**, *104*, 4569–4580. [CrossRef]
20. Abdoun, N.; El Assad, S.; Deforges, O.; Assaf, R.; Khalil, M. Design and security analysis of two robust keyed hash functions based on chaotic neural networks. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *11*, 2137–2161. [CrossRef]
21. Field, M.; Golubitsky, M. Symmetric Chaos: A pictorial exploration of an order imposed by symmetry within chaotic systems. *Comput. Phys.* **1990**, *4*, 470–479. [CrossRef]
22. Zhang, F.; Liang, Z.Y.; Yang, B.L.; Zhao, X.J.; Guo, S.Z.; Ren, K. Survey of design and security evaluation of authenticated encryption algorithms in the CAESAR competition. *Front. Inf. Technol. Electron. Eng.* **2018**, *19*, 1475–1499. [CrossRef]
23. Li, Z.; Bi, W.; Dong, X.; Wang, X. Improved conditional cube attacks on Keccak keyed modes with MILP method. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; pp. 99–127.
24. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G.; Van Keer, R. Keyak v2. CAESAR Submission. 2015.
25. Jean, J.; Nikolić, I.; Peyrin, T.; Seurin, Y. The Deoxys AEAD Family. *J. Cryptol.* **2021**, *34*, 31. [CrossRef]
26. Zhang, P.; Yuan, Q. Lightweight Authenticated Encryption Mode with Enhancing Security Guarantees. In Proceedings of the 2021 IEEE 6th International Conference on Computer and Communication Systems (ICCCS), Chengdu, China, 23–26 April 2021; pp. 689–694.
27. Pan, X.; Li, F. Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *J. Syst. Archit.* **2021**, *115*, 102075. [CrossRef]
28. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.* **2021**, *34*, 33. [CrossRef]
29. Rogaway, P. *Advances in Cryptology, Proceedings of the CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011*; Springer Science & Business Media: Boston, MA, USA, 2011; Volume 6841.
30. Abdoun, N.; El Assad, S.; Manh Hoang, T.; Deforges, O.; Assaf, R.; Khalil, M. Designing Two Secure Keyed Hash Functions Based on Sponge Construction and the Chaotic Neural Network. *Entropy* **2020**, *22*, 1012. [CrossRef] [PubMed]

31. Rogaway, P. Authenticated-encryption with associated-data. In Proceedings of the 9th ACM Conference on Computer and Communications Security 2002, Washington, DC, USA, 18–22 November 2002; pp. 98–107.
32. Siegenthaler, T. Decrypting a class of stream ciphers using ciphertext only. *IEEE Trans. Comput.* **1985**, *34*, 81–85. [CrossRef]
33. Lian, S.; Sun, J.; Wang, Z. Security analysis of a chaos-based image encryption algorithm. *Phys. A Stat. Mech. Its Appl.* **2005**, *351*, 645–661. [CrossRef]
34. Taha, M.A.; Assad, S.E.; Queudet, A.; Deforges, O. Design and efficient implementation of a chaos-based stream cipher. *Int. J. Internet Technol. Secur. Trans.* **2017**, *7*, 89–114. [CrossRef]
35. Biham, E.; Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptol.* **1991**, *4*, 3–72. [CrossRef]
36. Wu, Y.; Noonan, J.P.; Agaian, S. NPCR and UACI randomness tests for image encryption. *Cyber J. Multidiscip. J. Sci. Technol. J. Sel. Areas Telecommun. (JSAT)* **2011**, *1*, 31–38.
37. Mar, P.P.; Latt, K.M. New analysis methods on strict avalanche criterion of S-boxes. *World Acad. Sci. Eng. Technol.* **2008**, *2*, 899–903.
38. Wang, X.; Luan, D.; Bao, X. Cryptanalysis of an image encryption algorithm using Chebyshev generator. *Digit. Signal Process.* **2014**, *25*, 244–247. [CrossRef]
39. Xiao, D.; Liao, X.; Deng, S. One-way Hash function construction based on the chaotic map with changeable-parameter. *Chaos Solitons Fractals* **2005**, *24*, 65–71. [CrossRef]
40. Zhang, J.; Wang, X.; Zhang, W. Chaotic keyed hash function based on feedforward–feedback nonlinear digital filter. *Phys. Lett. A* **2007**, *362*, 439–448. [CrossRef]
41. Preneel, B. Analysis and Design of Cryptographic Hash Functions. Ph.D. Thesis, Katholieke Universiteit te Leuven, Leuven, The Netherlands, 1993.
42. Feistel, H. Cryptography and computer privacy. *Sci. Am.* **1973**, *228*, 15–23. [CrossRef]
43. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]
44. Barker, E.B.; Kelsey, J.M. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*; US Department of Commerce, Technology Administration, National Institute of Standards and Technology, Computer Security Division, Information Technology Laboratory: Washington, DC, USA, 2007.
45. Wu, Y.; Zhou, Y.; Saveriades, G.; Agaian, S.; Noonan, J.P.; Natarajan, P. Local Shannon entropy measure with statistical tests for image randomness. *Inf. Sci.* **2013**, *222*, 323–342. [CrossRef]
46. Song, C.Y.; Qiao, Y.L.; Zhang, X.Z. An image encryption scheme based on new spatiotemporal chaos. *Opt.-Int. J. Light Electron Opt.* **2013**, *124*, 3329–3334. [CrossRef]
47. Simplicio, M.A.; de Oliveira, B.T.; Barreto, P.S.; Margi, C.B.; Carvalho, T.C.; Naslund, M. Comparison of authenticated-encryption schemes in wireless sensor networks. In Proceedings of the 2011 IEEE 36th Conference on Local Computer Networks, Bonn, Germany, 4–7 October 2011; pp. 450–457.
48. Švenda, P. Basic Comparison of Modes for Authenticated-Encryption (IAPM, XCBC, OCB, CCM, EAX, CWC, GCM, PCFB, CS). 2016; Volume 35. Available online: https://www.fi.muni.cz/~xsvenda/docs/AE_comparison_ipics04.pdf (accessed on 10 August 2021).
49. Patel, M.; Venkatesan, S.; Weiner, D. Role assignment for data aggregation in wireless sensor networks. In Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), Niagara Falls, ON, Canada, 21–23 May 2007; Volume 2, pp. 390–395.