Hindawi
Security and Communication Networks
Volume 2017, Article ID 9040518, 15 pages
https://doi.org/10.1155/2017/9040518

Research Article

A Chaos-Based Authenticated Cipher with Associated Data

Je Sen Teh and Azman Samsudin

School of Computer Sciences, Universiti Sains Malaysia (USM), 11800 Penang, Malaysia

Correspondence should be addressed to Azman Samsudin; azman.samsudin@usm.my

Received 12 July 2016; Accepted 5 October 2016; Published 12 January 2017

Academic Editor: Pino Caballero-Gil

In recent years, there has been a rising interest in authenticated encryption with associated data (AEAD) which combines encryption and authentication into a unified scheme. AEAD schemes provide authentication for a message that is divided into two parts: associated data which is not encrypted and the plaintext which is encrypted. However, there is a lack of chaos-based AEAD schemes in recent literature. This paper introduces a new 128-bit chaos-based AEAD scheme based on the single-key Even-Mansour and Type-II generalized Feistel structure. The proposed scheme provides both privacy and authentication in a single-pass using only one 128-bit secret key. The chaotic tent map is used to generate whitening keys for the Even-Mansour construction, round keys, and random s-boxes for the Feistel round function. In addition, the proposed AEAD scheme can be implemented with true random number generators to map a message to multiple possible ciphertexts in a nondeterministic manner. Security and statistical evaluation indicate that the proposed scheme is highly secure for both the ciphertext and the authentication tag. Furthermore, it has multiple advantages over AES-GCM which is the current standard for authenticated encryption.

1. Introduction

Encryption and authentication of data are traditionally performed by two separate algorithms under the influence of different secret keys. This is known as generic composition which has three variations: *Encrypt-and-MAC*, *MAC-then-Encrypt,* and *Encrypt-then-MAC* [1]. Recently, there has been a rising interest in the field of authenticated encryption with associated data (AEAD) which combines encryption and authentication into a unified scheme. It deals with the cryptographic problem of sending a message that has to be entirely authenticated but divided into two parts: associated data (additional information such as packet headers) that must be sent in the clear and the actual plaintext that must be encrypted [2]. AEAD schemes are also more efficient and less prone to implementation errors, unlike generic composition [3]. This is because an AEAD scheme can be implemented without extensive cryptographic knowledge and is easy for interoperability [4]. As it only requires one key for both privacy and integrity, it saves on key bits and key setup time and reduces the risk of users selecting insecure parameters.

The Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [5] was introduced in 2014 to identify authenticated ciphers that offer advantages over AES-GCM, which has weak key problems and is susceptible to a cyclic attack [6]. CAESAR is currently in its second round of evaluation, with 30 candidate ciphers remaining. AEAD proposals for CAESAR need to have higher level of security and be as fast as AES-GCM or faster than AES-GCM with similar security level. The design of AEAD schemes is based on various constructs such as block ciphers, stream ciphers, and sponge constructions.

The aim of this paper is to construct an AEAD scheme based on chaos theory that is secure and efficient and fulfils the requirements set by CAESAR. For additional security, the AEAD scheme is designed to be able to take advantage of true random number generators (TRNG) to resist statistical-based cryptanalysis. Due to the existence of efficient and uniformly distributed software-based TRNGs, it is feasible to utilize true random numbers in cryptographic algorithms. These TRNGs take advantage of physical phenomena that occur within computing hardware such as multicore processors [7], graphics processing units [8], or hard disks [9] and quantifies them to generate nondeterministic numbers. As the inputs of AEAD schemes include secret and public message numbers, TRNGs can be implemented to generate secret message numbers to provide immunity to statistical-based cryptanalysis.

In this paper, a new AEAD scheme based on the chaotic tent map is proposed. It is designed based on the single-key Even-Mansour construction [10] and utilizes key-dependent random s-boxes. Both the whitening keys for the Even-Mansour scheme and s-boxes are generated based on chaos. The proposed AEAD scheme requires only a single-pass to produce both the authentication tag (MAC) and ciphertext. The cipher is evaluated for resistance against cryptanalytic attacks and its security is also analysed based on thorough statistical testing. Although not compulsory, the AEAD scheme can be implemented alongside a TRNG to provide immunity to statistical-based cryptanalysis.

The rest of this paper is structured as follows: Section 2 provides an introduction to AEAD schemes and chaotic maps that are vital to understanding the rest of the paper. Section 3 describes the proposed AEAD algorithm in detail, followed by Section 4 that analysed its security. The paper is concluded in Section 5 with some closing remarks.

2. Preliminaries

2.1. Authenticated Encryption with Associated Data. The following are inputs to an authenticated encryption algorithm along with their corresponding security requirements (I = Integrity, P = Privacy):

(i) Variable-length plaintext (ip)

(ii) Variable-length associated data (i)

(iii) q-bit secret message number (ip)

(iv) p-bit public message number/nonce (i)

(v) k-bit secret key

The output of an AEAD scheme consists of a ciphertext with the same length as the plaintext (except for schemes with secret message numbers which have extra blocks of ciphertext depending on the size of q) and an authentication tag or MAC of t bits. AEAD schemes need not support secret message numbers ($q = 0$) but require unique nonces for semantic security. The inputs to an authenticated decryption algorithm include the ciphertext, associated data, nonce, and secret key. The secret message number will be extracted from ciphertext. It outputs the plaintext if verification is successful; otherwise the algorithm outputs an error message.

2.2. Chaotic Maps. Chaotic maps are deterministic dynamical systems that have random-like behavior, sensitivity to slight changes in initial parameters, ergodicity, diffusion, and confusion characteristics. These qualities are analogous to requirements in cryptographic algorithms; thus chaotic maps have been used to design hash functions [11], encryption algorithms [12], key exchange protocols [13], and digital signature schemes [14]. Authenticated image encryption schemes based on chaos have been introduced by Bakhshandeh and Eslami [15] as well as Yang et al. [16]. However, they are both generic composition schemes with a distinct hash function and encryption algorithm. No chaos-based, unified AEAD schemes have been introduced in recent literature.

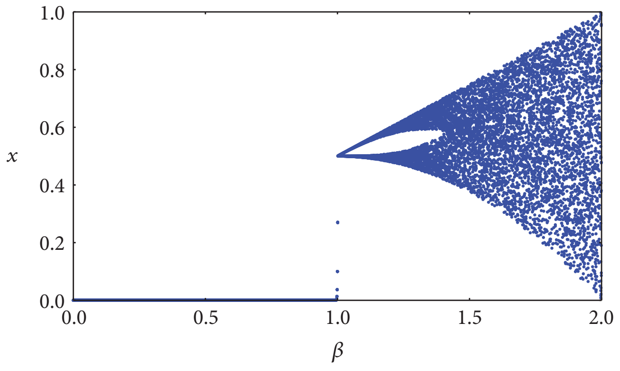


FIGURE 1: Tent map bifurcation diagram.

The chaotic tent map is used to design the proposed AEAD scheme as shown in (1), where β is the control parameter that falls within the range of $[0, 2]$. In addition to being a fast and simple map, the tent map is selected because it has been widely studied and applied in the field of chaotic cryptography. The behavior of the chaotic map evolves from periodic to aperiodic as β increases from 0 to 2 as shown in the tent map's bifurcation diagram in Figure 1. The proposed AEAD sets $\beta = 1.99999$ for all tent map iterations.

$$c(x_{t+1}) = \begin{cases} \beta x_t & 0 \le x_t < 0.5 \\ \beta(1 - x_t), & 0.5 \le x_t \le 1. \end{cases} \tag{1}$$

Recent literature has shown that unimodal chaotic maps such as the tent map are susceptible to initial condition/parameter estimation techniques using statistical tools [17, 18]. Another problem of digital chaotic systems is dynamical degradation that occurs due to finite precision, which leads to short cycle lengths [19]. These problems can be addressed by using chaotic perturbation which changes the trajectory of the chaotic map's orbit by modifying its initial condition using external values. The proposed AEAD scheme takes advantage of chaotic perturbation in its design as discussed in Section 3.

In software, chaotic map iterations involve real numbers that are usually represented by floating point variables. However, floating point operations are slower than binary operations. For faster speed and reproducibility, fixed point (FxP) representation [20] is used in the proposed AEAD scheme. A 32-bit FxP variable consists of the two most significant bits (MSB) that represent integers before the radix point and the 30 least significant bits (LSB) that represent fractional bits.

3. Design Specifications

3.1. Parameters and Notations. All integers stated in the following algorithms are in hexadecimal format. Prior to modification by the nonce or secret keys, the initial values for

Table 1: Block shuffle π and inverse block shuffle π^{-1}.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\pi(x)$	5	0	1	4	7	c	3	8	d	6	9	2	f	a	b	e
$\pi^{-1}(x)$	1	2	b	6	3	0	9	4	7	a	d	e	5	8	f	c

```
(1) **Input:** 64-bit secret key values $k_0$ and $k_1$.
(2) **Output:** Initialized variables $x, y, p, q, r_0, r_1$.
(3)—
(4) //Initialize variables
(5) $x \leftarrow (82dcbe4e \oplus k_0) \wedge \Phi,\ y \leftarrow (49a2372c \oplus k_1) \wedge \Phi$
(6) $p \leftarrow (49a2372c \oplus (k_0 \gg 30)), q \leftarrow (945ae69b \oplus (k_1 \gg 30))$
(7) $r_0 \leftarrow (3 \boxplus_8 (k_0 \gg 62)), r_1 \leftarrow (3 \boxplus_8 (k_1 \gg 62))$
(8) **for** $i = 0$ **to** 3 **do**
(9)     $c_{50}(x)$
(10)    $c_{50}(y)$
(11)    $x \leftarrow (x \oplus (p \boxplus_{32} \mathrm{rcon}_{4i})) \wedge \Phi$
(12)    $p \leftarrow \mathrm{rotl}^{32}_{r_0}(p \boxplus_{32} \mathrm{rcon}_{4i+1})$
(13)    $y \leftarrow (y \oplus (q \boxplus_{32} \mathrm{rcon}_{4i+2})) \wedge \Phi$
(14)    $q \leftarrow \mathrm{rotl}^{32}_{r_1}(q \boxplus_{32} \mathrm{rcon}_{4i+3})$
(15) **end for**
```

Algorithm 1: Initialization of the Chaos-based RNG.

all registers are arbitrarily chosen random numbers. The proposed 128-bit AEAD scheme has the following parameters:

(i) 128-bit secret key, $\kappa \in \{0,1\}^{128}$

(ii) 64-bit nonce, $\eta \in \{0,1\}^{64}$

(iii) 128-bit secret message number, $\tau \in \{0,1\}^{128}$

(iv) Associated data, $A \in \{0,1\}^*$

(v) Message, $M \in \{0,1\}^*$

(vi) 128-bit tag, $T \in \{0,1\}^{128}$

The following are notations used in this paper:

(i) Length of a string, x, denoted by $|x|$

(ii) $\oplus$: Bitwise XOR

(iii) $\wedge$: Bitwise AND

(iv) $x \parallel y$: Bitwise concatenation of x and y

(v) $\boxplus_n$: n-bit modular addition

(vi) $\mathrm{rotl}^n_r(x)$: Bitwise left rotation by r positions on a n-bit variable, x where $n = \{8, 32, 64\}$

(vii) $x \gg y$: Bitwise right shift of x by y bits

(viii) $s(x)$: substitution function using an 8-bit s-box, s

(ix) $c_i(x)$: performing i-iterations of the chaotic map with an initial value x

(x) $\lfloor x \rfloor$: floor function to map a real number x to the previous largest integer

(xi) FP(x): converting the FxP representation to real number

(xii) Φ: mask for the FxP fractional bits

Because the AEAD design involves chaotic map iterations, round constants are used in several parts of the algorithms to prevent weak key problems. Weak keys can initialize the chaotic map's initial condition to zero, which makes the system nonrandom and insecure. These round constants are taken from the list of prime numbers: $\mathrm{rcon}_i = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131\}$, where $0 \le i < 32$ is the index/round. The proposed AEAD scheme also uses a block shuffle π and its inverse as shown in Table 1.

3.2. Random S-Box Generation. The composition method [21] is used to generate chaos-based s-boxes for the AEAD scheme. The composition of two 8-bit permutations, $f(x)$ and $g(x)$, is denoted by $h(x) = f(g(x))$, $\forall x \in I$, where I is the set of all 8-bit integers. First, a set S consisting of four chaos-based s-boxes is generated by permuting the AES s-box. To obtain the final s-box, a composition of four s-boxes from S is used, where the s-boxes are selected based on randomly generated indices (repetitions are allowed). To generate the numbers used to permute AES and also as indices for the s-box composition, a chaos-based RNG is used. The initialization process and the RNG algorithm are as shown in Algorithms 1 and 2, respectively, where x and y are chaotic map initial conditions that are constantly modified by perturbation values, p and q.

The initialization process equally divides the secret key bits $\kappa = k_1 \parallel k_0$ into four registers and two rotation variables such that all key bits are used only once. During the initialization, round constants are used to ensure that values of κ that initialize all registers to zero will result in nonzero starting values for the RNG. A total of 200 iterations for each tent map are performed for transient elimination, although 48 iterations have been deemed sufficient [22].

```
(1) Input: 32-bit initial values $x$ and $y$, 32-bit perturbation values $p$ and $q$, 8-bit
    rotation values $r_0$ and $r_1$.
(2) Output: 32-bit random number $R$.
(3) —
(4) $c_{50}(x)$
(5) $c_{50}(y)$
(6) $R \leftarrow \lfloor \mathrm{FP}(x) \rfloor \oplus \lfloor \mathrm{FP}(y) \rfloor$
(7) $x \leftarrow (x \oplus p) \wedge \Phi$
(8) $p \leftarrow \mathrm{rotl}_{r_0}^{32}(p)$
(9) $y \leftarrow (y \oplus q) \wedge \Phi$
(10) $q \leftarrow \mathrm{rotl}_{r_1}^{32}(q)$
```

Algorithm 2: Chaos-based RNG.

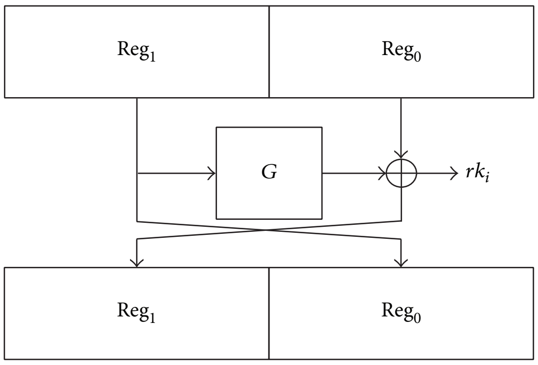


Figure 2: Key Schedule Feistel Network.

Each 8-bit s-box, $s_j \in S$ for $0 \le j < 4$, is first initialized as the AES s-box. Next, Algorithm 2 is used to generate v_i^j for $0 \le i < 256$ for each s_j. Each s-box is then permuted as follows:

$$\begin{aligned} s_j[i] &= s_j\left[v_i^j\right], \\ s_j\left[v_i^j\right] &= s_j[i]. \end{aligned} \tag{2}$$

To obtain the final s-box, s_f, four additional values are generated by Algorithm 2. A four-modulo operation is applied on these values which are then used as s-box indices, I_i for $0 \le i < 4$. The final s-box is a composition of the four s-boxes indexed by I: $s_f(x) = s_{I_3}(s_{I_2}(s_{I_1}(s_{I_0}(x))))$. As s_f depends only on κ, it can be precomputed by both parties. s_f is later modified by τ in the actual encryption process.

3.3. Key Schedule Algorithm. The key scheduling algorithm generates 20 round keys from the secret key κ. As they do not depend on η or τ, the keys can also be precomputed. The algorithm is based on the Feistel network, whereby the secret key is first divided into two halves, $\kappa = k_1 \parallel k_0$. k_0 and k_1 are then XOR-ed into two registers that have initial values of $\mathrm{reg}_0 = f492e42164c7fb0d$ and $\mathrm{reg}_1 = f683d0ad038f17d$. The tent map's initial condition is set to $x = 0$. One round of the key scheduling function is shown in Figure 2, where i is the round number.

The G-function is described in Algorithm 3. The network is executed 16 times for transient elimination without producing any keys. Starting from round number $i = 0, 20$ round keys are then produced from 20 rounds of the network. Excluding transient elimination rounds, each round key is generated from a total 640 iterations of the chaotic tent map. The value of O is modified 64 times, where, each time, it is subjected to a four-bit rotation to ensure that all 64 bits are modified by x in 16 passes. reg_1 is rotated by 15 bits each time to ensure that its upper and lower bits are all used in the chaotic iterations.

3.4. Masking Algorithm. The proposed AEAD scheme is based on the single-key Even-Mansour construction that has been proven to have the same level of security as the original two-key construction [10]. 16 eight-bit mask registers are first initialized based on the secret key and nonce and then modified by a chaotic function. The values of these registers are used as the prewhitening and postwhitening masking keys for each block of data. The initialization of the mask registers and variables for the chaotic function are summarized in Figure 3, where $\kappa = k_1 \parallel k_0$.

After the mask and chaotic variables are initialized, the masking algorithm is iterated eight times for transient elimination and to diffuse the key bits into all registers. The masking algorithm is as shown in Algorithm 4 where registers (x, y, p, q) are involved in chaotic perturbation. These registers are then used to modify the mask registers, λ. Round constants are only used in the transient elimination rounds to generate nonzero initial mask values. The mask registers are later modified by τ in the encryption process.

3.5. Algorithm Description. The overall design of the AEAD is based on the single-key Even-Mansour scheme [10] and a Type-II generalized Feistel structure (GFS) [23]. There are two main functions, *aead_encrypt* and *aead_decrypt*. *aead_encrypt* encrypts M and τ and also generates a tag, T, to verify A, M, and τ. *aead_decrypt* decrypts C to obtain M and τ and then generates T to verify the authenticity of A, M, and τ. If verification fails, the decrypted message is not released to the recipient. The *aead_encrypt* algorithm is shown in Figure 4, where a is the number of A blocks, m is the number of M blocks, α represents intermediate tags, $\sum_M$

```
(1) **Input:** 64-bit register $\text{reg}_1$, 32-bit initial condition $x$, round number $i$.
(2) **Output:** 64-bit output, $O$.
(3) —
(4) $O \leftarrow 0$
(5) **for** $j = 0$ **to** $63$ **do**
(6)     $x \leftarrow x \oplus (\text{reg}_1 \wedge \Phi)$
(7)     $c_{10}(x)$
(8)     $O \leftarrow O \oplus x$
(9)     $O \leftarrow \text{rotl}_4^{64}(O)$
(10)    $\text{reg}_1 \leftarrow \text{rotl}_{15}^{64}(\text{reg}_1)$
(11) **end for**
(12) $O \leftarrow O \oplus \text{rcon}_i$
```

Algorithm 3: Key schedule G-function.

```
(1) **Input:** 8-bit mask registers $\lambda_j$ for $0 \le j < 16$, 32-bit initial values $x$ and $y$, 32-bit perturbation
    values $p$ and $q$, round/block number $i$.
(2) **Output:** Modified 8-bit mask registers $\lambda_j$ for $0 \le j < 16$
(3) —
(4) $x \leftarrow c_1(x)$, $y \leftarrow c_1(y)$
(5) **if** Transient Elimination **then**
(6)     $x \leftarrow x \oplus (p \boxplus_{32} \text{rcon}_{4i})$
(7)     $y \leftarrow y \oplus (q \boxplus_{32} \text{rcon}_{4i+1})$
(8)     $p \leftarrow p \oplus (y \boxplus_{32} \text{rcon}_{4i+2})$
(9)     $q \leftarrow q \oplus (x \boxplus_{32} \text{rcon}_{4i+3})$
(10) **else**
(11)    $x \leftarrow x \oplus p$
(12)    $y \leftarrow y \oplus q$
(13)    $p \leftarrow p \oplus y$
(14)    $q \leftarrow q \oplus x$
(15) **end if**
(16) **for** $j = 0$ **to** $3$ **do**
(17)    $\lambda_j \leftarrow \lambda_j \boxplus_8 ((x \gg (8j)) \wedge FF)$
(18)    $\lambda_{j+4} \leftarrow \lambda_{j+4} \boxplus_8 ((y \gg (8j)) \wedge FF)$
(19)    $\lambda_{j+8} \leftarrow \lambda_{j+8} \boxplus_8 ((p \gg (8j)) \wedge FF)$
(20)    $\lambda_{j+12} \leftarrow \lambda_{j+12} \boxplus_8 ((q \gg (8j)) \wedge FF)$
(21) **end for**
(22) **for** $j = 0$ **to** $15$ **do**
(23)    $\lambda_{\pi(j)} \leftarrow \lambda_j$
(24) **end for**
(25) $x \leftarrow x \wedge \Phi$, $y \leftarrow y \wedge \Phi$
```

Algorithm 4: Mask algorithm.

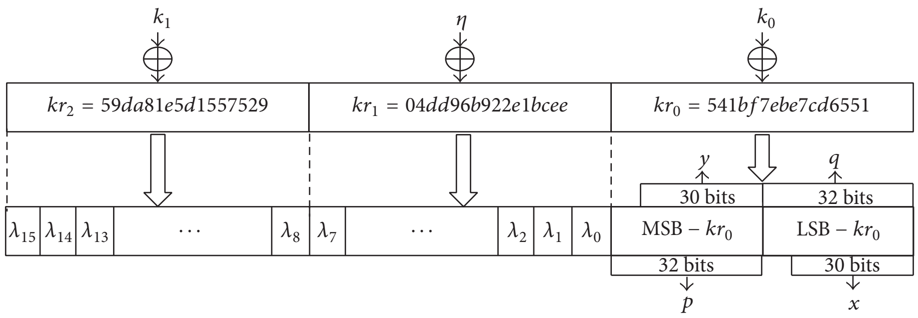

k_1
η
k_0
$kr_2 = 59da81e5d1557529$
$kr_1 = 04dd96b922e1bcee$
$kr_0 = 541bf7ebe7cd6551$
y
q
30 bits
32 bits
λ_{15} λ_{14} λ_{13} ⋯ λ_8 λ_7 ⋯ λ_2 λ_1 λ_0
MSB – kr_0
LSB – kr_0
32 bits
30 bits
p
x

Figure 3: Masking Algorithm Initialization.

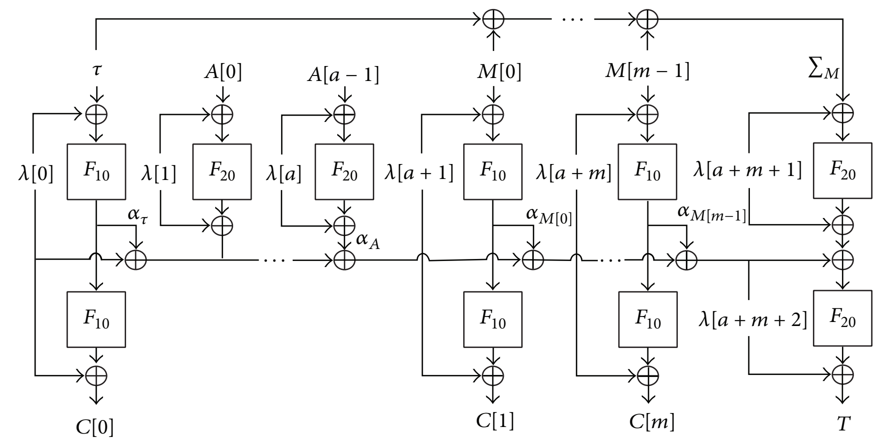
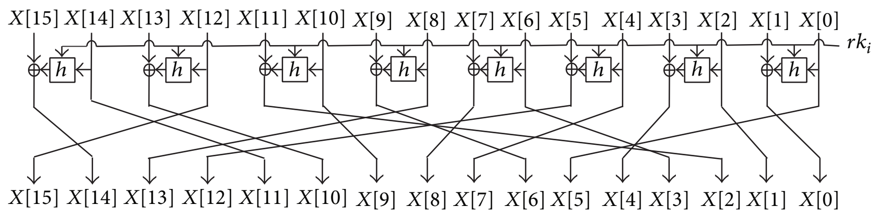


Figure 4: Encryption Algorithm.



X[15] X[14] X[13] X[12] X[11] X[10] X[9] X[8] X[7] X[6] X[5] X[4] X[3] X[2] X[1] X[0]
rk_i
h
X[15] X[14] X[13] X[12] X[11] X[10] X[9] X[8] X[7] X[6] X[5] X[4] X[3] X[2] X[1] X[0]

Figure 5: AEAD F-function.

is the checksum, and $\lambda[i]$ is the set of mask registers for block-i of data. The *aead_decrypt* algorithm has the same structure as *aead_encrypt*. However the inverse block shuffle π^{-1} and reversed round keys are used to decrypt C and to generate the intermediate tags, $\alpha_{M[*]}$.

3.5.1. Round Function. The round function, F_j, is executed 20 times for all data blocks where j denotes the number of rounds. The same 20 round keys generated by the key scheduling algorithm are used for all blocks. When processing message blocks ($M[*]$), F_{20} is divided into two halves, $F_{20} = F_{10} \cdot F_{10}$ where the output of the first 10 rounds is used as the intermediate tag $\alpha_{M[*]}$. The round function is based on the Type-II GFS as shown in Figure 5. The 64-bit round key rk_i is divided into 8 bytes, where its most significant byte is fed into the left most h-function. The h-function consists of two operations, XOR of the round key and substitution:

$$h(x) = s_f\left[X[d] \oplus rk_i^e\right], \quad (3)$$

where d and e are byte indexes of the data block and round key, respectively.

3.5.2. Modification of S-Box and Mask. For further security, τ is used to modify both s_f and λ prior to processing the first block of data. To modify s_f, two more indexes for s-box composition are obtained from τ, I_τ^0, and I_τ^1. I_τ^0 is obtained by XOR-ing each of the eight most significant bytes of τ whereas I_τ^1 is obtained by XOR-ing each of the eight least significant bytes. The final s-box is computed by composing two additional s-boxes: $s_{\text{final}}(x) = s_{I_\tau^1}(s_{I_\tau^0}(s_f(x)))$. The modification of λ registers is performed by XOR-ing each byte of τ to each byte of λ.

3.5.3. AEAD Mode. The encryption and tag generation process is as follows:

(1) Using κ, generate rk_i for $0 \le i < 20$ and s_f.

(2) Initialize λ based on κ and η.

(3) Encrypt τ to obtain $C[0]$ and α_τ.

(4) Generate s_{final} and modify λ using τ (Section 3.5.2).

(5) If $|A|$ and $|M|$ are not 128-bit multiples, pad A and M with a single bit of "1," followed by all "0" until the required length.

(6) Encrypt $A[j]$ for $0 \le j < a$ and XOR results to obtain α_A.

(7) Encrypt $M[j]$ to obtain $\alpha_{M[j]}$ and $C[j+1]$, where $0 \le j < m$.

(8) Compute $\sum_M$. Encrypt $\sum_M$ and XOR the result with α_τ, α_A, and $\alpha_{M[*]}$. Perform one final encryption to obtain T.

The decryption and tag verification process is as follows:

(1) Using κ, generate rk_i for $0 \le i < 20$ and s_f.

(2) Initialize λ based on κ and η.

(3) Decrypt $C[0]$ to obtain τ and α_τ.

(4) Generate s_{final} and modify λ using τ (Section 3.5.2).

(5) If $|A|$ is not a 128-bit multiple, pad A with a single bit of "1" followed by all "0" until the required length.

(6) Encrypt $A[j]$ for $0 \leq j < a$ and XOR results to obtain α_A.

(7) Decrypt $C[j+1]$ to obtain $\alpha_{M[j]}$ and $M[j]$, where $0 \leq j < m$.

(8) Compute $\sum_M$. Encrypt $\sum_M$ and XOR the result with α_τ, α_A, and $\alpha_{M[*]}$. Perform one final encryption to obtain T_{verify}.

(9) Compare T_{verify} with T. If they do not match, the plaintext is not released to the recipient and an error symbol, Γ, is output.

3.5.4. MAC-Only Mode. The proposed AEAD scheme can also be used solely as a MAC. In the MAC-only mode, the algorithm is the same except that ciphertext blocks need not be computed. In addition, the secret message number is unused and is set to $\tau = 0$.

3.6. Features. Chaos-based algorithms require real number computation which is usually performed using floating point representation. FxP representation is used to ensure that the proposed AEAD scheme is not platform-specific. In addition, it contributes to better performance as basic binary operations can be used on FxP variables which require fewer clock cycles than floating point operations.

The proposed cipher has several features that are advantageous for AEAD designs. Firstly, each block of data can be processed in parallel. The only overhead is the iteration of the masking algorithm which has to be performed based on the block number. It is an online, one-pass cipher whereby the encryption of one block does not depend on other blocks. The verification algorithm is highly nonce-misuse resistant due to the randomly generated τ. The proposed AEAD scheme also has out-of-order verification; whereby if any blocks are permuted or out of order, it will result in a different MAC.

To achieve high security against cryptanalytic attacks, users have the option of using a TRNG to generate τ. A TRNG depends on physical phenomena; therefore τ becomes nondeterministic in nature. Each plaintext is then randomly mapped to different ciphertexts. However, anyone with the correct secret key is able to extract τ to decrypt and verify the message successfully regardless of how τ was generated. Although the randomization of τ contributes to strong security characteristics as described in Section 4.5.4, the proposed AEAD scheme becomes nonincremental in nature. The entire associated data or message needs to be recomputed if one block is changed because τ changes upon each encryption.

4. Security Analysis

This section examines the security of all components of the proposed AEAD scheme. For all tests, inputs are set to zero ($\kappa = 0, \eta = 0, \tau = 0$) unless stated otherwise. As it is difficult to provide mathematical proof for real number-based cryptographic algorithms, the AEAD scheme is evaluated mainly using statistical means.

4.1. Preliminary Statistical Testing. Before other evaluation methods are used, statistical test suites such as NIST SP 800-22 [24], DIEHARD [25], and ENT [26] are used as preliminary tests for majority of the AEAD components. The minimum/maximum passing ratio (PR) for NIST and DIEHARD for a significance level of 0.01 is calculated based on (4), where N is the number of samples, as follows:

$$\text{PR} = 0.99 \pm 3 \times \sqrt{0.99 \times \frac{0.01}{N}}. \tag{4}$$

The summary of inputs required for each test suite is listed below.

(i) NIST: 1000-Mbit file generated with $\kappa = 0$, $\eta = 0$, $\tau = 0$, divided into 1000 samples

(ii) DIEHARD: 50 87.5-Mbit files generated with $\kappa = \text{rand}()$, $\eta = 0$, $\tau = 0$; each DIEHARD suite has 18 tests with multiple P values that are each taken as individual samples

(iii) ENT: 87.5-Mbit file generated with $\kappa = 0$, $\eta = 0$, $\tau = 0$.

The minimum P value to indicate uniformity for each NIST test is 0.0001 whereas the DIEHARD tests require P values in the range of $[0.01, 0.99]$. The min/max PR for the NIST and DIEHARD test suites are listed in Table 6 of the Appendix. As for the ENT test suite, a number sequence fails if its results stray too far from the ideal values listed in Table 7 of the Appendix. Passing all three test suites indicates that the number sequence is pseudorandom with no obvious statistical defects.

4.2. S-Box Evaluation. Random s-boxes are generated based on a chaotic number sequence which is tested using the NIST, DIEHARD, and ENT test suites. It passes all three as shown in Tables 8, 12, and 16 of the Appendix. In the following experiments, s-boxes are produced for $\kappa = \text{rand}()$, $\eta = 0$, $\tau = 0$ where their characteristics such as the avalanche effect, strict avalanche criterion (SAC), linear and differential probability, nonlinearity, and keyspace are examined.

4.2.1. Avalanche and Strict Avalanche Criterion. To exhibit the avalanche effect, one-half of the output bits should change on average whenever a single input bit is toggled. To determine if an 8-bit s-box fulfils this requirement, the 2^8 possible inputs are first divided into 2^7 pairs, (X, X_i), which differ only in bit i. The 8-bit avalanche vectors, V_i, are then calculated for all $(0 \leq i < 8)$ as follows:

$$V_i = s(X) \oplus s(X_i). \tag{5}$$

For 10000 random s-boxes all possible input pairs are tested. Results indicate that 50.1899% of each V_i bits are equal to one when an input differs in only bit i.

To satisfy the SAC, each individual output bit should change with a probability of 50% when a single input bit is toggled. Similar to the previous test, 2^8 possible inputs are first divided into 2^7 pairs, (X, X_i), which differ only in bit i. The 8-bit avalanche vectors, V_i are then calculated for all $(0 \leq i < 8)$. For 10000 random s-boxes and all possible input pairs, the average probability that a bit location in V_i is equal to one when an input differs in only bit i is 50.1993%.

Both results show that the s-boxes generated by the proposed AEAD scheme have strong avalanche effect and fulfil the SAC. Satisfying the SAC also implies that the s-box is a complete function, whereby each output bit is dependent on all input bits [27].

4.2.2. Differential Distribution and Linear Approximation. For 10000 randomly generated s-boxes, the differential distribution [28] and linear approximation [29] tables are computed. These tables consist of the probabilities of differential and linear characteristics for various input/output combinations that can be used in differential and linear attacks. The worst case differential and linear probabilities are $2^{-4.67807}$ and $2^{-2.91254}$, respectively. These probabilities are sufficient to resist cryptanalytic attacks which are described in Section 4.5.

4.2.3. Nonlinearity and Key Sensitivity. Let $\mathbb{F}_2^n$ be the n-dimensional vector space over the field $\mathbb{F}_2$. The nonlinearity of a function $F : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ is defined by

$$\mathrm{NL}(F) = \left(2^{n-1} - \frac{1}{2}\right) \max_{a\in\mathbb{F}_2^n, b\in\mathbb{F}_2^n} \left| \sum_{x\in\mathbb{F}_2^n} (-1)^{b\cdot F(x)+a\cdot x} \right|. \qquad (6)$$

A perfectly nonlinear function has a nonlinearity of $\mathrm{NL}(F) = 2^{n-1} - 2^{n/2-1}$, which is 120 for an 8-bit s-box. Out of 10000 randomly generated s-boxes, the average nonlinearity is 92.6646, which is 77.22% nonlinear. The generated s-boxes have lower nonlinearity than the AES s-box ($\mathrm{NL}(F_{\mathrm{AES}}) = 112$). However, unlike a static s-box, attackers cannot linearly approximate a randomized s-box without knowledge of the keys used to generate it. Therefore, slight changes to the secret key should generate different s-boxes. For all 128-bit secret keys with hamming weight of one, the proposed AEAD scheme generates unique s-boxes. In addition, for 2000 incrementally generated s-boxes ($\kappa = 0$ to $\kappa = 7d0$), all s-boxes are also unique. This shows that the random s-box algorithm is highly sensitive to slight changes to κ.

4.3. Mask Evaluation. The output of the masking algorithm is first tested using the NIST, DIEHARD, and ENT test suites. The 128-bit mask values generated by the AEAD scheme successfully pass all three as shown in Tables 9, 13, and 17 of the Appendix.

4.3.1. Mask Cycle Length. The cycle length of the masking algorithm must be sufficiently large to avoid attacks that can take advantage of any mask value cycles. Theoretically, the cycle length is limited by the precision of the fixed point representation used in chaotic iterations. This amounts to $2^{30} - 1$, as 30 bits are used to represent the fractional bits. This is a conservative bound, as chaotic perturbation and chaotic coupling further extend the cycle length. Because there are two chaotic maps used to modify the mask registers, the total cycle length is the least common multiple of their cycle lengths. Based on this lower bound, the proposed AEAD scheme will be able to support messages of approximately $2^{60} - 1$ blocks before a mask cycle occurs.

4.4. Key Schedule Evaluation. The key schedule algorithm generates only 20 round keys for a secret key. Therefore, data for the statistical test suites are generated using incremental secret keys (incremented from $\kappa = 0$) except for the DIEHARD test which still uses randomly generated secret keys. Results in Tables 10, 14, and 18 of the Appendix indicate that the key schedule algorithm passes all test suites.

4.4.1. Avalanche and Strict Avalanche Criterion. The key schedule algorithm is analysed for avalanche effect and SAC similar to the s-box evaluation in Section 4.2.1. The avalanche vectors V_i are calculated using (5), where 10000 randomly generated pairs of X, X_i are used as inputs. To depict the avalanche effect, one-half of each round key should change on average whenever a single input key bit is toggled. The percentages of V_i bits that are equal to one when the input key differs in only bit i are 50.0403%, 49.99%, 50.0719%, 49.9422%, 49.9228%, 49.9047%, 50.0417%, 49.92%, 49.9942%, 49.9925%, 49.8973%, 50.0503%, 50.1373%, 50.0075%, 49.9709%, 49.9975%, 50.0342%, 50.0542%, 49.9628%, and 49.9153% for each of the 20 rounds, respectively. This has an overall average of 49.9924% per round.

To fulfil the SAC, each round key bit should change with a probability of 50% when a single input key bit is toggled. For each of the 20 rounds, the average probabilities that a bit location in V_i is equal to one when the input key differs in only bit i are 49.9998%, 49.9891%, 50.0914%, 49.8953%, 49.9978%, 49.9583%, 49.9866%, 50.0108%, 49.9941%, 49.9044%, 49.9095%, 49.9452%, 50.0287%, 50.0473%, 50.0302%, 50.0439%, 50.0852%, 50.017%, 50.0633%, and 49.9419%. This has an overall average of 49.997% per round.

Results show that the key schedule algorithm has strong avalanche effect and fulfils the SAC. This implies that the key schedule algorithm is a complete function, where all round key bits are dependent on all input secret key bits.

4.5. Encryption Algorithm Security. This section analyses the security of the AEAD scheme against various cryptanalytic attacks. A lower bound of security against these attacks is first determined with $\tau = 0$. Then, additional security provided by using a TRNG to generate τ is discussed. Prior to other security analyses, the ciphertext of the AEAD scheme is tested using statistical suites with results in Tables 11, 15, and 19.

4.5.1. Differential and Linear Cryptanalysis. The security bound for differential and linear cryptanalysis can be determined based on the minimum number of active s-boxes for a differential characteristic (AS_D^r) and linear characteristic (AS_L^r), where r is the number of rounds. The number of AS is dependent on the type of block shuffle used in the round function. In an early design, the number 10 block shuffle from [30] was used in the proposed AEAD scheme, supposedly with $\mathrm{AS}_D^{20}/\mathrm{AS}_L^{20} = 44$. However, a branch and bound algorithm was used and found that $\mathrm{AS}_D^{20}/\mathrm{AS}_L^{20} = 37$.

Table 2: List of active S-boxes.

Round, r	AS_D^r, AS_L^r
1	0
2	1
3	2
4	3
5	4
6	6
7	8
8	11
9	14
10	18
11	22
12	24
13	27
14	30
15	32
16	35
17	36
18	39
19	41
20	44

Instead, an isomorphic shuffle to the number 10 shuffle was used, adopted from [31] with AS_D^r and AS_L^r for $1 \le r \le 20$ shown in Table 2.

The maximum differential and linear probabilities for the AEAD scheme's s-boxes are $P_D = 2^{-4.67807}$ and $P_L = 2^{-2.91254}$, respectively as mentioned in Section 4.2.2. The probability for differential and linear distinguishers are calculated using

$$P^r_{\text{diff-dist}} = {P_D}^{AS_D^r},$$
$$P^r_{\text{lin-dist}} = \frac{1}{2} + \left(2^{AS_L^r-1} \times {P_L}^{AS_L^r}\right). \tag{7}$$

For $r = 14$ and $r = 16$, $P^{14}_{\text{diff-dist}} = 2^{-140.3421}$ and $P^{16}_{\text{lin-dist}} = 1/2 + 2^{-67.9389}$, where $2^{-67.9389}$ is the linear bias, denoted by ε_{lin}. The number of plaintexts required for a differential and linear distinguishing attack is then $1/P^{14}_{\text{diff-dist}} = 2^{140.3421}$ and $1/(\varepsilon_{\text{lin}})^2 = 2^{135.8778}$, respectively, which are both more than the available 2^{128}. Therefore, the full 20 rounds have a safe security margin against differential and linear cryptanalysis.

4.5.2. Algebraic Attack. In an algebraic attack, s-boxes are described by an overdefined system of algebraic equations that hold true with a probability of one [32]. The use of random s-boxes increases the resistance of a cipher against this attack [33]. The proposed AEAD scheme uses key-dependent s-boxes which are further randomized by τ; therefore it is difficult to obtain a system of equations for these s-boxes. Without these equations, an algebraic attack is not feasible. The proposed cipher is thus resistant to algebraic attacks.

4.5.3. Timing Attack. Timing attacks exploit operations with data-dependent execution time to reveal the secret key [34, 35]. In the proposed AEAD scheme, the secret key is used to generate s-boxes and round keys and also to initialize the mask register. The remaining operations consist of Bitwise operations that have constant execution time. Therefore, the proposed cipher is resistant to timing attacks because its encryption speed is independent of the key value.

4.5.4. Security with True Random Secret Message Number. If τ is generated by a nondeterministic TRNG, the cipher is able to resist a wide range of statistical-based cryptanalytic attacks. A nondeterministic τ ensures that multiple encryptions of even the same (A, M) pair will involve different s-boxes and masking values. This leads to different (C, T) outputs. However, a legitimate recipient can easily decrypt and verify these ciphertext-MAC pairs as long as the secret key is correct.

Randomized s-boxes prevent any cryptanalytic attacks that rely on approximating the behavior of the s-box, such as differential, linear, and algebraic attacks. Even if an approximation of the s-box is obtained for a particular plaintext-ciphertext pair, it is only applicable to that specific pair. The cipher is also immune to attacks that rely on selecting input differences that cancel out the effect of encryption keys such as the differential attack, truncated differential attack, and impossible differential attack. This is because the masking keys are modified by the truly random τ, making them nondeterministic for each plaintext encryption. For a differential characteristic to hold, τ values for a plaintext pair have to match, which occurs with a probability of 2^{-128} for a uniform TRNG.

Key recovery attacks based on an r-round statistical distinguisher involve the encryption of multiple plaintexts for $r + g$ rounds to obtain their corresponding $(r + g)$-round ciphertexts. Attackers then guess subkey bits, decrypt the ciphertext for g rounds, and verify their guess based on the r-round statistical distinguisher. If τ is nondeterministic, attackers will not be able to obtain the correct subkey using this method as each plaintext is encrypted based on different s-boxes and masking keys. In practice, the statistical distinguisher only holds if two plaintexts are encrypted with same τ, which again occurs with a probability of 2^{-128} for a uniform TRNG.

As previously mentioned, generating τ using a TRNG makes the cipher misuse resistant. If the same nonce is used twice, the resulting MAC and ciphertext are still secure as long as τ is not repeated for the same (A, M, η) 3-tuple. In short, implementing the proposed AEAD scheme with a TRNG or even a hybrid RNG will strengthen its immunity against statistical-based cryptanalysis as well as user misuse.

4.6. MAC Security. This section analyses the statistical quality of the authentication tag produced by the proposed AEAD scheme. In the following experiments, the input parameters are set to $\kappa = 0$, $\eta = 0$, $\tau = 0$, whereby only A and M vary.

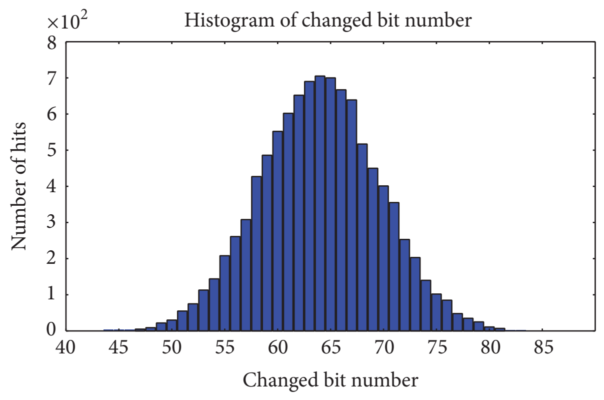


Figure 6: Distribution histogram of X_i.

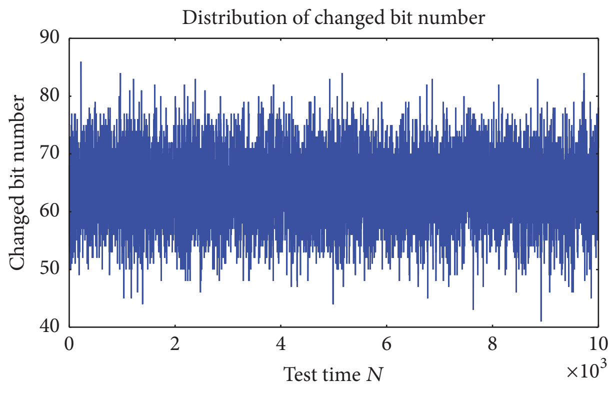


Figure 7: Distribution plot of X_i.

4.6.1. Statistical Analysis of Diffusion and Confusion. In this test, an (A, M) pair is randomly generated and its corresponding tag, T, is computed. Next, a random bit within the (A, M) pair is toggled and its corresponding tag, T', is computed. T and T' values are compared and the number of changes at each bit location is stored as X_i. The following equations are then computed:

Mean Changed Bit Number

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i. \tag{8}$$

Mean Changed Probability

$$P = \frac{\overline{X}}{128} \times 100\%. \tag{9}$$

Standard Variance of the Changed Bit Number

$$\Delta X = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(X_i - \overline{X}\right)^2}. \tag{10}$$

Table 3: Statistical results for 128-bit hash values of the proposed hash function.

Param/N	1024	2048	10000	Mean
$\overline{X}$	64.0010	64.0078	63.9968	64.0019
P (%)	50.0008	50.0061	49.9975	50.0015
ΔX	5.6291	5.6755	5.6291	5.6446
ΔP (%)	4.3977	4.4340	4.3978	4.4098
$X_{\min}$	47	45	45	45.67
$X_{\max}$	83	83	86	84

Table 4: Number of hits.

Hits	0	1	2	3	4	5	6
Theoretical	9416	572	12	0	0	0	0
Experimental	9428	558	13	1	0	0	0

Table 5: Absolute difference.

Results	Minimum	Maximum	Mean
Theoretical	—	—	1360
Experimental	574	2352	1362.55

Standard Variance

$$\Delta P = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\frac{X_i}{128} - P\right)^2} \times 100\%, \tag{11}$$

where N is the number of trials. The test is performed for $N = 1024/2048/10000$ and the results are tabulated in Table 3. The distribution plot and histogram of X_i for $N = 10000$ are shown in Figures 7 and 6. Results show that the MAC values have a mean changed bit number, $\overline{X}$, and mean changed probability, P, that are near-ideal (64 bits and 50%). Low standard variance values depict strong diffusion and confusion capabilities. In Figures 7 and 6, X_i is shown to be evenly and normally distributed centering on 64 bits. These near-ideal statistical results indicate that the slightest change to the A or M will produce a different authentication tag.

4.6.2. Analysis of Collision Resistance. Similar to Section 4.6.1, MACs for an (A, M) pair and a slightly modified $(A, M)'$ pair are produced and stored in ASCII format. The two MACs are compared and the number of hits (equal ASCII characters in the same position) are counted. Next, the absolute difference, D, between the two MACs is calculated as follows:

$$D = \sum_{i=1}^{N}\left|d\left(e_i\right) - d\left(e_i'\right)\right|, \tag{12}$$

where e_i and e_i' are the ith entry of the original and modified MAC, while the function $d(*)$ converts the entry to its equivalent decimal value. The test results for 10000 trials are compared against theoretical values (see [36] for calculation of theoretical values) in Tables 4 and 5. Results indicate that the proposed AEAD scheme has strong collision resistance as the experimental results are near-ideal.

Table 6: Passing Ratios for NIST and DIEHARD.

Test name	Samples	Min. PR	Max. PR
NIST (except random excursions)	1000	0.981	0.999
NIST, random excursions	625	0.978	1.000
DIEHARD, birthday spacings	500	0.977	1.000
DIEHARD, overlapping 5-permutation	100	0.961	1.000
DIEHARD, binary rank 31 × 31 matrices	50	0.949	1.000
DIEHARD, binary rank 32 × 32 matrices	50	0.949	1.000
DIEHARD, binary rank 6 × 8 matrices	1300	0.982	0.998
DIEHARD, bitstream	1000	0.981	0.999
DIEHARD, overlapping-pairs-sparse-occupancy	1150	0.981	0.999
DIEHARD, overlapping-quadruples-sparse-occupancy	1400	0.9822	0.998
DIEHARD, DNA spacings	1550	0.983	0.997
DIEHARD, Count-the-1s on a Stream of Bytes	100	0.961	1.000
DIEHARD, count-the-1s on specific bytes	1250	0.982	0.998
DIEHARD, parking lot	550	0.978	1.000
DIEHARD, minimum distance	50	0.949	1.000
DIEHARD, 3D-spheres	1050	0.984	0.999
DIEHARD, squeeze	50	0.949	1.000
DIEHARD, overlapping sums	550	0.978	1.000
DIEHARD, runs	200	0.969	1.000
DIEHARD, craps	100	0.961	1.000

Table 7: Ideal values for ENT test suite.

Test name	Ideal value
Entropy	8.0
Chi-square	50%
Arithmetic mean	127.5
Monte Carlo value for π	3.141592653
Serial correlation coefficient	9

Table 8: NIST: S-box chaotic sequence.

Test name	P value	Passing ratio	Result
Frequency	0.771469	0.993	Pass
Block frequency	0.759756	0.989	Pass
Cumulative sums	0.122325	0.990	Pass
Runs	0.179584	0.990	Pass
Longest run	0.446556	0.989	Pass
Rank	0.568739	0.993	Pass
FFT	0.689019	0.991	Pass
Nonoverlapping templates	0.224821	0.981	Pass
Overlapping templates	0.146152	0.984	Pass
Universal	0.011383	0.982	Pass
Approximate entropy	0.042808	0.987	Pass
Random excursions	0.173679	0.987	Pass
Random excursions variant	0.711017	0.981	Pass
Serial	0.803720	0.981	Pass
Linear complexity	0.660012	0.990	Pass

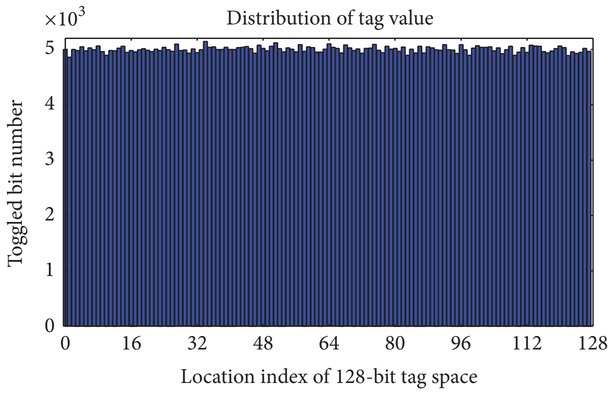


Figure 8: Distribution of MAC.

4.6.3. MAC Distribution. To resist statistical attacks, a MAC has to be evenly distributed at each bit position. Analysis of the MAC distribution starts with generating a random (A, M) pair and its modified version with one toggled bit, $(A, M)'$. The MAC of both messages is compared, and the number of changes at each bit location is counted. For $N = 10000$, the minimum, maximum, and mean changed bit number are 4877, 5150, and 5002.3, respectively, with the distribution plot shown in Figure 8. The mean changed bit number is close to the theoretical value of 5000, where the maximum distance from ideal for each changed bit value is 150. This indicates that proposed AEAD scheme produces evenly distributed MAC values as depicted in Figure 8.

Table 9: NIST: Masking algorithm.

Test Name	P value	Passing ratio	Result
Frequency	0.390721	0.991	Pass
Block frequency	0.530120	0.987	Pass
Cumulative sums	0.259616	0.989	Pass
Runs	0.406499	0.992	Pass
Longest run	0.307077	0.989	Pass
Rank	0.066882	0.991	Pass
FFT	0.556460	0.981	Pass
Nonoverlapping templates	0.238035	0.982	Pass
Overlapping templates	0.676615	0.984	Pass
Universal	0.834308	0.992	Pass
Approximate entropy	0.915317	0.991	Pass
Random excursions	0.582671	0.979	Pass
Random excursions variant	0.422281	0.993	Pass
Serial	0.875539	0.984	Pass
Linear complexity	0.890582	0.989	Pass

Table 10: NIST: Key schedule.

Test Name	P value	Passing ratio	Result
Frequency	0.085587	0.991	Pass
Block frequency	0.008629	0.989	Pass
Cumulative sums	0.719747	0.990	Pass
Runs	0.143686	0.990	Pass
Longest run	0.467322	0.989	Pass
Rank	0.363593	0.993	Pass
FFT	0.015816	0.987	Pass
Nonoverlapping templates	0.540204	0.981	Pass
Overlapping templates	0.065230	0.988	Pass
Universal	0.401199	0.991	Pass
Approximate entropy	0.077131	0.996	Pass
Random excursions	0.098397	0.982	Pass
Random excursions variant	0.884892	0.984	Pass
Serial	0.446556	0.989	Pass
Linear complexity	0.249284	0.990	Pass

4.7. Advantage over AES-GCM. The proposed scheme has several advantages over AES-GCM which is currently the official standard for authenticated encryption [37]. Firstly, the proposed AEAD scheme does not have any identified weak keys unlike AES-GCM [6]. It is also resistant to nonce-misuse, providing full privacy and integrity even if nonce is repeated, unlike AES-GCM which requires unique nonce values for security [37]. Most software-based AES implementations are susceptible to cache timing attacks [35], whereas the proposed cipher has a constant time implementation due to simple binary operations. AES-GCM's security bound for integrity is equal to half the tag length [6] whereas the proposed AEAD scheme is able to provide full 128-bit security for integrity. Furthermore, the proposed AEAD scheme also has the flexibility to be implemented with a TRNG for immunity to statistical-based attacks.

Table 11: NIST: ciphertext.

Test name	P value	Passing ratio	Result
Frequency	0.206629	0.989	Pass
Block frequency	0.207730	0.997	Pass
Cumulative sums	0.206629	0.990	Pass
Runs	0.373625	0.995	Pass
Longest run	0.016149	0.987	Pass
Rank	0.388990	0.989	Pass
FFT	0.954930	0.987	Pass
Nonoverlapping templates	0.454053	0.982	Pass
Overlapping templates	0.467322	0.985	Pass
Universal	0.129620	0.985	Pass
Approximate entropy	0.358641	0.990	Pass
Random excursions	0.866173	0.986	Pass
Random excursions variant	0.956262	0.991	Pass
Serial	0.887645	0.992	Pass
Linear complexity	0.271619	0.986	Pass

Table 12: DIEHARD: S-box chaotic sequence.

Test name	Passing ratio	Result
Birthday spacings	0.978	Pass
Overlapping 5-permutation	0.970	Pass
Binary Rank 31×31 matrices	0.980	Pass
Binary Rank 32×32 matrices	1.00	Pass
Binary Rank 6×8 matrices	0.985	Pass
Bitstream	0.982	Pass
Overlapping-pairs-sparse-occupancy	0.982	Pass
Overlapping-quadruples-sparse-occupancy	0.983	Pass
DNA spacings	0.985	Pass
Count-the-1s on a stream of bytes	0.980	Pass
Count-the-1s on specific bytes	0.983	Pass
Parking lot	0.978	Pass
Minimum distance	0.960	Pass
3D-spheres	0.990	Pass
Squeeze	0.960	Pass
Overlapping sums	0.985	Pass
Runs	0.980	Pass
Craps	0.990	Pass

In terms of performance, the speed of AES-GCM was compared against the proposed cipher on a computer with an Intel i7-4700MQ processor and 8 GB of RAM. The implementation of AES-GCM was taken from the Crypto++ Library 5.6.2 [38]. The proposed AEAD scheme has a speed of 2401.73 Megabits/second compared to 1601.4 Megabits/second of AES-GCM, approximately 1.5 times faster. The algorithm can still be further optimized for improved performance.

5. Conclusion

This paper introduced a new chaos-based authenticated encryption with associated data (AEAD) scheme. It is based on the single-key Even-Mansour construction and Type-II

Table 13: DIEHARD: masking algorithm.

Test Name	Passing Ratio	Result
Birthday spacings	0.982	Pass
Overlapping 5-permutation	0.970	Pass
Binary rank 31 × 31 matrices	1.000	Pass
Binary rank 32 × 32 matrices	0.960	Pass
Binary rank 6 × 8 matrices	0.982	Pass
Bitstream	0.982	Pass
Overlapping-pairs-sparse-occupancy	0.983	Pass
Overlapping-quadruples-sparse-occupancy	0.984	Pass
DNA spacings	0.984	Pass
Count-the-1s on a stream of bytes	0.990	Pass
Count-the-1s on specific bytes	0.984	Pass
Parking lot	0.980	Pass
Minimum distance	0.960	Pass
3D-spheres	0.985	Pass
Squeeze	0.980	Pass
Overlapping sums	0.980	Pass
Runs	0.990	Pass
Craps	0.990	Pass

Table 14: DIEHARD: key schedule.

Test name	Passing ratio	Result
Birthday spacings	0.982	Pass
Overlapping 5-permutation	0.970	Pass
Binary rank 31 × 31 matrices	0.980	Pass
Binary rank 32 × 32 matrices	1.000	Pass
Binary rank 6 × 8 matrices	0.982	Pass
Bitstream	0.982	Pass
Overlapping-pairs-sparse-occupancy	0.983	Pass
Overlapping-quadruples-sparse-occupancy	0.982	Pass
DNA spacings	0.983	Pass
Count-the-1s on a stream of bytes	0.970	Pass
Count-the-1s on specific bytes	0.982	Pass
Parking lot	0.980	Pass
Minimum distance	0.980	Pass
3D-spheres	0.984	Pass
Squeeze	1.000	Pass
Overlapping sums	0.987	Pass
Runs	0.975	Pass
Craps	0.970	Pass

generalized Feistel with random s-boxes. The chaotic tent map was used to generate Even-Mansour whitening keys, round keys, and s-boxes used in the Feistel round functions. Chaotic perturbations were included in the proposed design to overcome weaknesses of unimodal chaotic maps. For improved performance, the fixed point representation was used for all real number operations. Results of rigorous statistical testing have shown that the proposed AEAD scheme has no observable statistical defects. In addition, it is resistant to cryptanalytic attacks such as the differential, linear, algebraic, and timing attacks. To increase its immunity against all statistical-based attacks, the AEAD can be implemented alongside a true random number generator to map messages to various ciphertexts in an unpredictable manner. It also has better performance than AES-GCM which is the current standard in authenticated encryption, with a throughput of approximately 2401.73 Megabits/second. In short, the

Table 15: DIEHARD: ciphertext.

Test name	Passing ratio	Result
Birthday spacings	0.982	Pass
Overlapping 5-permutation	0.970	Pass
Binary rank 31 × 31 matrices	0.980	Pass
Binary rank 32 × 32 matrices	0.980	Pass
Binary rank 6 × 8 matrices	0.982	Pass
Bitstream	0.987	Pass
Overlapping-pairs-sparse-Occupancy	0.982	Pass
Overlapping-quadruples-sparse-occupancy	0.984	Pass
DNA Spacings	0.985	Pass
Count-the-1s on a stream of bytes	0.970	Pass
Count-the-1s on specific bytes	0.983	Pass
Parking lot	0.978	Pass
Minimum distance	0.980	Pass
3D-spheres	0.983	Pass
Squeeze	0.960	Pass
Overlapping sums	0.980	Pass
Runs	0.970	Pass
Craps	1.000	Pass

Table 16: ENT: S-box chaotic sequence.

Test name	Test value	Result
Entropy	7.999984	Pass
Chi-square	55.13%	Pass
Arithmetic mean	127.5205	Pass
Monte Carlo value for π	3.140525649	Pass
Serial correlation coefficient	−0.000300	Pass

Table 17: ENT: masking algorithm.

Test name	Test value	Result
Entropy	7.999984	Pass
Chi-square	55.08%	Pass
Arithmetic mean	127.4735	Pass
Monte Carlo value for π	3.141910973	Pass
Serial correlation coefficient	−0.000407	Pass

Table 18: ENT: vey schedule.

Test Name	Test value	Result
Entropy	7.999984	Pass
Chi-square	62.61	Pass
Arithmetic mean	127.4953	Pass
Monte Carlo value for π	3.140523556	Pass
Serial correlation coefficient	−0.000304	Pass

Table 19: ENT: Ciphertext.

Test name	Test value	Result
Entropy	7.999984	Pass
Chi-square	55.43%	Pass
Arithmetic mean	127.5039	Pass
Monte Carlo value for π	3.141705895	Pass
Serial correlation coefficient	0.000032	Pass

proposed chaos-based AEAD scheme is a viable alternative to AES-GCM in terms of providing both privacy and integrity in a unified scheme.

Appendix

See Tables 6–19.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This work has been supported by Fundamental Research Grant Scheme (FRGS, 203/PKOMP/6711427) funded by the Ministry of Higher Education of Malaysia (MOHE). A special thanks to WeiJian Teng for his help in proofreading this paper.

References

[1] M. Bellare and C. Namprempre, "Authenticated encryption: relations among notions and analysis of the generic composition paradigm," in *Advances in Cryptology—ASIACRYPT 2000: 6th International Conference on the Theory and Application of Cryptology and Information Security Kyoto, Japan, December 3–7, 2000 Proceedings*, T. Okamoto, Ed., vol. 1976, pp. 531–545, Springer, Berlin, Germany, 2000.

[2] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS '02)*, pp. 98–107, November 2002.

[3] N. Ferguson and B. Schneier, "A cryptographic evaluation of IPSec," Tech. Rep., Counterpane Internet Security, 2000.

[4] M. Bellare, P. Rogaway, and D. Wagner, "The EAX mode of operation," in *Fast Software Encryption: 11th International Workshop (FSE 2004)*, B. Roy and W. Meier, Eds., pp. 389–407, Springer, Berlin, Germany, 2004.

[5] Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR), https://competitions.cr.yp.to/caesar-call.html.

[6] M. J. O. Saarinen, "Cycling attacks on GCM, GHASH and other polynomial MACs and hashes," in *Fast Software Encryption: 19th International Workshop, FSE 2012*, A. Canteaut, Ed., pp. 216–225, Springer, Berlin, Germany, 2012.

[7] S. Muller, "CPU time jitter based non-physical true random number generator," 2014.

[8] J. S. Teh, A. Samsudin, M. Al-Mazrooie, and A. Akhavan, "GPUs and chaos: a new true random number generator," *Nonlinear Dynamics*, vol. 82, no. 4, pp. 1913–1922, 2015.

[9] D. Davis, R. Ihaka, and P. Fenstermacher, "Cryptographic randomness from air turbulence in disk drives," in *Advances in Cryptology—CRYPTO '94*, Y. Desmedt, Ed., vol. 839 of *Lecture Notes in Computer Science*, pp. 114–120, Springer, Berlin, Germany, 1994.

[10] O. Dunkelman, N. Keller, and A. Shamir, "Minimalism in cryptography: the even-mansour scheme revisited," in *Advances in Cryptology—EUROCRYPT 2012: 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15–19, 2012. Proceedings*, vol. 7237 of *Lecture Notes in Computer Science*, pp. 336–354, Springer, Berlin, Germany, 2012.

[11] J. S. Teh, A. Samsudin, and A. Akhavan, "Parallel chaotic hash function based on the shuffle-exchange network," *Nonlinear Dynamics*, vol. 81, no. 3, pp. 1067–1079, 2015.

[12] M. K. Mandal, M. Kar, S. K. Singh, and V. K. Barnwal, "Symmetric key image encryption using chaotic Rossler system," *Security and Communication Networks*, vol. 7, no. 11, pp. 2145–2152, 2014.

[13] H.-Y. Lin, "Chaotic map-based three-party authenticated key agreement," *Security and Communication Networks*, vol. 7, no. 12, pp. 2469–2474, 2014.

[14] P. Muthukumar, P. Balasubramaniam, and K. Ratnavelu, "Synchronization of a novel fractional order stretch-twist-fold (STF) flow chaotic system and its application to a new authenticated encryption scheme (AES)," *Nonlinear Dynamics*, vol. 77, no. 4, pp. 1547–1559, 2014.

[15] A. Bakhshandeh and Z. Eslami, "An authenticated image encryption scheme based on chaotic maps and memory cellular automata," *Optics and Lasers in Engineering*, vol. 51, no. 6, pp. 665–673, 2013.

[16] H. Yang, K.-W. Wong, X. Liao, W. Zhang, and P. Wei, "A fast image encryption and authentication scheme based on chaotic maps," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, no. 11, pp. 3507–3517, 2010.

[17] X. Wu, H. Hu, and B. Zhang, "Parameter estimation only from the symbolic sequences generated by chaos system," *Chaos, Solitons & Fractals*, vol. 22, no. 2, pp. 359–366, 2004.

[18] D. Arroyo, G. Alvarez, J. Amigó, and S. Li, "Cryptanalysis of a family of self-synchronizing chaotic stream ciphers," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, no. 2, pp. 805–813, 2011.

[19] S. Li, X. Mou, Y. Cai, Z. Ji, and J. Zhang, "On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision," *Computer Physics Communications*, vol. 153, no. 1, pp. 52–58, 2003.

[20] R. Yates, *Fixed-Point Arithmetic: An Introduction*, Digital Signal Labs, 2013, http://www.digitalsignallabs.com/fp.pdf.

[21] D. Lambić and M. Živković, "Comparison of random s-box generation methods," *Publications de l'Institut Mathématique*, vol. 93(107), no. 113, pp. 109–115, 2013.

[22] J. A. Oteo and J. Ros, "Double precision errors in the logistic map: statistical study and dynamical interpretation," *Physical Review E*, vol. 76, no. 3, Article ID 036214, 2007.

[23] Y. Zheng, T. Matsumoto, and H. Imai, "On the construction of block ciphers provably secure and not relying on any unproved hypotheses," in *Advances in Cryptology—CRYPTO' 89 Proceedings*, G. Brassard, Ed., pp. 461–480, Springer, New York, NY, USA, 1990.

[24] A. Rukhin, J. Soto, and J. Nechvatal, “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” NIST Special Publication 800-22, National Institute of Standards and Technology, 2010.

[25] G. Marsaglia, DIEHARD: A battery of tests of Randomness, 1996, http://stat.fsu.edu/pub/diehard/.

[26] J. Walker, *ENT Program*, 2008, http://www.fourmilab.ch/random.

[27] A. F. Webster and S. E. Tavares, “On the design of s-boxes,” in *Advances in Cryptology—CRYPTO ’85 Proceedings*, vol. 218 of *Lecture Notes in Computer Science*, pp. 523–534, Springer, Berlin, Germany, 2000.

[28] E. Biham and A. Shamir, “Differential cryptanalysis of DES-like cryptosystems,” in *Advances in Cryptology—CRYPT0 ’90*, A. J. Menezes and S. A. Vanstone, Eds., vol. 537 of *Lecture Notes in Computer Science*, pp. 2–21, Springer, Berlin, Germany, 1991.

[29] M. Matsui, “Linear cryptanalysis method for DES cipher,” in *Advances in Cryptology—EUROCRYPT ’93*, pp. 386–397, Springer, Berlin, Germany, 1994.

[30] T. Suzaki and K. Minematsu, “Improving the generalized feistel,” in *Fast Software Encryption: 17th International Workshop, FSE 2010, Seoul, Korea, February 7–10, 2010, Revised Selected Papers*, vol. 6147 of *Lecture Notes in Computer Science*, pp. 19–39, Springer, Berlin, Germany, 2010.

[31] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, “TWINE: a lightweight block cipher for multiple platforms,” in *Selected Areas in Cryptography: 19th International Conference, SAC 2012*, L. R. Knudsen and H. Wu, Eds., pp. 339–354, Springer, Berlin, Germany, 2013.

[32] N. T. Courtois and J. Pieprzyk, “Cryptanalysis of block ciphers with overdefined systems of equations,” in *Advances in Cryptology—ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security Queenstown, New Zealand, December 1–5, 2002 Proceedings*, Y. Zheng, Ed., vol. 2501 of *Lecture Notes in Computer Science*, pp. 267–287, Springer, Berlin, Germany, 2002.

[33] N. T. Courtois, *General Principles of Algebraic Attacks and New Design Criteria for Components of Symmetric Ciphers*, vol. 3373 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2005.

[34] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” in *Advances in Cryptology—CRYPTO ’96*, N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113, Springer, Berlin, Germany, 1996.

[35] D. Bernstein, “Cache-timing attacks on AES,” 2005, http://cr.yp.to/papers.html#cachetiming.

[36] A. Akhavan, A. Samsudin, and A. Akhshani, “A novel parallel hash function based on 3D chaotic map,” *EURASIP Journal on Advances in Signal Processing*, vol. 2013, article 126, 2013.

[37] M. Dworkin, “Recommendation for block cipher modes of operation: galois/counter mode (gcm) and gmac,” Tech. Rep., National Institute of Standards and Technology (NIST), 2007.

[38] Crypto++ library, http://www.cryptopp.com/.