Received: October 11, 2022. Revised: December 14, 2022.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Design a Lightweight Authentication Encryption Based on Stream Cipher and Chaotic Maps with Sponge Structure for Internet of Things Applications

Rana Saad Mohammed[1]*

[1]*Department of Computer Science, Education college, Mustansiriyah University, Baghdad, Iraq*
*Corresponding author's Email: drranasaad@uomustansiriyah.edu.iq

Abstract: The design of a cipher system is essential and required with the development of the technology era. The internet of things concept is modern technology requiring lightweight, secure algorithms to investigate the requirement of the light device. Furthermore, IoT devices have limited resources (IoT devices should minimize battery life, limited memory size, and response time). A stream cipher system is faster than a block cipher system and has an easy feature with hardware and software implementation. This paper designs a lightweight authentication encryption algorithm based on stream cipher and chaotic maps with sponge structure for IoT applications. Currently, the suggested approach uses text data for encryption and authorization. Additionally, the suggested algorithm will be able to use it with image data in the future. Analysis results of the proposed algorithm based on NIST randomness tests, execution time, memory space, and functional features. The results are compared with other previous works which used sponge structure (e.g. Ascon, Elephant, ISAP, Photon-Beetle, and Xoodyak, etc.). Randomness tests show that the proposed system is random and secure with investigate all the functional features except the parallelization feature which will work it in the future. And also it has good fast speed and less memory space.

Keywords: Lightweight, Authentication encryption, CAESAR competition, NIST-LW competition, Stream cipher, Chaotic map, Linear feedback shift register (LFSR), NonLinear feedback shift register (NFSR), Sponge structure, Internet of things (IoT).

1. Introduction

The increasing demand for an internet of things (IoT) in e-commerce and e-government. This demand requires the efficiency and security of cryptography algorithms. Cryptography faces the challenges of providing lightweight features in speed and memory with higher security. Cryptography has divided into stream cipher and block cipher. Stream cipher is easier to implement in hardware and software than block cipher, and it also features quick encryption and decryption operations with little to no error propagation [1, 2].

Different designs of stream ciphers depend on their structure: LFSR, NFSR, FCSR, PANAMA, random shuffled, ARX, sponge structural, block cipher based stream cipher, block cipher work mode based stream cipher, fully homomorphic encryption systems, and provable secure.

A lightweight stream cipher design is required to be applicable in IoT applications. And also, creating authenticated encryption algorithms based on stream cipher is an open problem [3].

Stream ciphers can be designed using permutations or sponge structures. It allows for any amount of data in the input and output and uses a broad random permutation.

Sponge structure becomes excellent flexibility and can provide authentication property without adding an authentication module. However, there is a problem with cryptanalysis studies about this structure, and it is better to develop it in the future.

A group of international cryptologic researchers created the competition for authenticated encryption: Security, applicability, and robustness (CAESAR) to promote the creation of authenticated encryption systems. In January 2013, the competition was announced, and the final portfolio was released in February 2019.

The final portfolio announced by the CAESAR committee are Ascon and ACORN as lightweight applications.

In March 2021, the national institute of standards and technology (NIST) announced ten finalists as ASCON, elephant, GIFT-COFB, grain128-AEAD, ISAP, photon-beetle, romulus, sparkle, TinyJambu, and Xoodyak.

These finalists have different structure. This paper focused on sponge structure only as ASCON, elephant, ISAP, photon-beetle, and Xoodyak.

In [4] classified the above winner finalists and the other different algorithms that submitted into the CAESAR and the NIST-LW competitions to evaluate the degree to which they provide the relevant functional features and security-related characteristics.

The six functional features are parallelization, online, inverse-free, single-pass, N. misuse resist, and Lightweight functional features as described in the following section 5 with Table 5 summarizes the results. The results show these different design of sponge structures have weakened for investigate all of the required functional features. All of them used fixed permutations [3].

This paper tries to investigate most of these features by design a lightweight authentication encryption algorithm based on stream and chaotic maps with sponge structure.

The category of this paper can be as follows: introduction in section 1, related work in section 2, fundamental components of the proposed system in section 3, proposed authentication encryption system in section 4, the analysis result in section 5, and conclusion in the last section.

2. Related works

There are different designs of sponge structure [4] as SpongeWrap [5], APE [6], CBEAM [7], Sp-AELM [8], Lsap [9], Beetle [10], Spookchain [11], T-sponge [12], TETsponge [13], CTR[14], ISAP[15], Beetle [16], Cyclist [17], Duplex [19, 20].

Each of them has advantages and drawbacks. The work [4] shows these drawbacks depend on the investigate of functional features. The functional features are parallelization, online, inverse-free, single-pass, N. misuse resist, and lightweight functional features.

The design SpongeWrap [5] investigated *single pass* functional feature only. APE [6] investigated *N. misuse resist and ligthwight* functional features only. CBEAM [7] investigated *ligthwight* functional feature only. Sp-AELM [8] nvestigated *Online, ligthwight, and RUP security* functional features only.

The design Lsap [9] not investigated any of functional features. Beetle [10] investigated Inverse-free, Single-pass and lightweight functional features only. Spookchain [11] investigated Online, lightweight, and BBB security functional features only.

The design T-sponge [12] not investigated any of functional features. TETsponge [13] investigated single-pass functional features only. CTR [14] investigated Parallelizable, Inverse-free, and Lightweight functional features only. ISAP [15] investigated Inverse-free, single-pass, Lightweight, and RUP Security functional features only.

The design Beetle [16] investigated Inverse-free, single-pass, and Lightweight functional features only. Cyclist [17, 18] investigated Inverse-free, single-pass, Lightweight, and RUP Security functional features only. Duplex [19] investigated Inverse-free functional feature only.

The design duplex (ASCON) [20] investigated Online, inverse-free, Single-pass, N. misuse resist, and Lightweight functional features only.

There are two categories of attacks [21-25] on Ascon: forgery and key recovery attacks. Key recovery attacks concentrate on the initialization step, whereas forgery attacks concentrate on the finalization step. The other papers modified Ascon as in [26-29].

3. Fundamental components of the proposed system

This paper describes the essential components of the proposed system as follows:

3.1 Feedback shift register (FSR) [3]

Among the many parts of a stream cipher is the feedback shift register (FSR). It is now separated into linear feedback shift registers (LFSR) and nonlinear feedback shift registers (NFSR). LFSR divides into bit-oriented and word-oriented.

Programmers, mathematicians, engineers, and other professionals all use the term finite state machine (FSM) to refer to a mathematical model of any system with a finite number of conditional states of existence. It is an abstract machine that has a finite number of states and can only ever be in one of them at once. Additionally, the FSM has the ability to switch between states in response to some inputs. The states, initial state, and inputs of an FSM are listed.

In NFSR, a type of stream cipher takes NFSR as an essential component. In order to offer security, it makes use of both nonlinear feedback (polynomial) and nonlinear output. FSM, nonlinear filter, or nonlinear combiner are techniques used with chaos map and LFSR to get nonlinear properties. And also, NFSR uses a nonlinear function to get nonlinear properties.

This paper takes some chaotic functions that have random and nonlinear properties.

3.2 Chaotic map

The study of chaotic systems, which have unpredictable appearances and irregular behaviors, is a subfield of mathematics. With sensitive beginning conditions, ergodic and regulated parameters, and irregular-like behavior similar to traditional crypto-systems' characteristics, chaotic crypto-systems had secured correspondences [30][31].

There are papers insert chaotic maps into cipher system as in [32-38].

Arnold map can be as follows:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & a \\ b & ab+1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} mod(n) \quad (1)$$

Where:

n=1,2, 3, …, N-1 (2)

x_n, y_n: are the position of samples in the N×N matrix.

x_{n+1}, y_{n+1} is the transformed position after the cat map.

a, b: are the positive integers of control parameters.

Henon map can be represented as follows:

$$x_{n+1} = 1 + by_n - ax_n^2 \quad (3)$$

$$y_{n+1} = x_n \quad (4)$$

It can be chaotic if a=1.4 and b=0.3.

While **Standard map** can be represented as follows:

$$x_{n+1} = x_n + k \sin y_n \mod 2\pi \quad (5)$$

$$y_{n+1} = y_n + x_{n+1} \mod 2\pi \quad (6)$$

Where [0,2π] is the range for x_n and y_n. When the parameter k has a value greater than or equal to 18.9 (k ≥18.9), this map is regarded as chaotic.

3.3 SPECK block cipher for initialization step [39]

Speck is a lightweight block cipher released by national security agency (NSA) in June 3013. Speck has performance implementation in hardware and software. The operations of speck are Bitwise XOR (⊕), addition modulo 2^n (⊞), and Left and Right Circular Shifts, S^j and S^{-j}, by j bits, respectively, are all contained in one round of Speck.

The following algorithm of Speck cipher:

Algorithm 1: Speck cipher (SpeckAlgorithm1)

```
Definition:
n = word size (16, 24, 32, 48, or 64)
m = number of key words
T = number of round
if n = 16 then (α, β) = (7,2)
    otherwise (α, β) = (8,3)
x,y = plaintext words
l[m-2].. l[0],k[0] = key words
------------------ key expansion ------------------
for i = 0..T-2
l[i+m-1]  = (k[i] ⊞ $S^{-\alpha}$l[i]) ⊕ i
k[i+1] =  $S^{\beta}$k[i] ⊕ l[i+m-1]
end for
------------------- encryption ---------------------
for i = 0..T-1
x = ($S^{-\alpha}$x ⊞ y) ⊕ k[i]
y = $S^{\beta}$y ⊕ x
end for
```

4. Proposed authenticated encryption based stream cipher

The proposed design is using LFSR with FSM using sponge structure and chaotic maps to recover the problems reviewed in the previous section. For example, forgery and key recovery attacks break the Ascon by focusing on the initialization phase, finalization phase, S-box, and permutation.

This section, we design an authentication encryption system that enhances the initialization and finalization phases and uses dynamic substitution and permutation.

Fig. 1 shows the proposed authentication encryption system based on a sponge structure. It consists of four phases: The initialization, the Association data, the Encryption, and the finalization. Each phase constructs from the round function of speck block cipher, LFSR, FSM, and random chaotic map.

While Fig. 2 shows the proposed initialization phase and Fig. 3 shows the general proposed function f of these phases. This function has three inputs and three outputs. Each phase has different inputs and outputs but the same function f.

The following figures and algorithms depict the work of the proposed encryption authentication system in detail.

4.1 Initialization phase

At first, get the key and initial vector (IV), each with 128 bits. Then, apply the round function of a speck cipher ten times. Each output of a round saves as a state in LFSR. The outcomes of the initialization phase are:

$$\text{Ioutput1}=S_t, \tag{7}$$

$$\text{Ioutput2}=S_{t+5}\oplus \text{ the output (y) of StanardMap of equation 6,} \tag{8}$$

$$\text{Ioutput3}=S_{t+8} \tag{9}$$

They are go to the next phase (Association data). The LFSR feedback by Ioutput2 to update St+9 and shifting the LFSR.

Algorithm 2: Initialization phase (InitAlgorithm2)

```
Input: Key 128 bit, IV 128 bit
Output: Ioutput1, Ioutput2, Ioutput3
Begin
For t=1 to 10
  S[t]=0
For r= 1 to 32
  Round[r]=SpeckAlgorithm1(Key,IV)
For s= 23 to 32
  S[s-22]=round[s]
Ioutput1=S[1]
Ioutput2=S[6] ⊕ StanardMap(y)
Ioutput3=S[9]
For t= 1 to 9
  S[t]=S[t+1]
 S[10]= Ioutput2
End
```

4.2 Association data phase

In the second phase, get the three outputs of the previous step (Ioutput1, Ioutput2, Ioutput3), each with 128 bits. Update Ioutput1 by XORing with 128 bits of association data (Ai). Where 1<= i <= n. If length A>128, then padding and split into blocks (n), each block 128 bits. Then, apply the proposed function fi (n) times. The outcomes of the Association data phase (Aoutput1, Aoutput2, Aoutput3) go to the next phase (Encryption) if i>n.

Algorithm 3: Associarion data (AssoAlgorithm3)

```
Input: A[1…n], Ioutput1, Ioutput2, Ioutput3
Output:Aoutput1[1…n],Aoutput2[1…n],
Aoutput3[1…n]
Begin
Aoutput1[1]= Ioutput1; Aoutput2[1]= Ioutput2;
Aoutput3[1]= Ioutput3;
For i=1 to n
Input1[i]= A[i] ⊕ Aoutput1[i]
Input2[i]= Aoutput2[i]
Input3[i]= Aoutput3[i]
FAlgorithm11(Input1[i],Input2[i],Input3[i],
Aoutput1[i+1], Aoutput2[i+1], Aoutput3[i+1])
End for
End
```

4.3 Encryption phase on the sender side

In the third phase, get the three outputs of the previous step (Aoutput1, Aoutput2, Aoutput3), each with 128 bits. Update Aoutput1 by XORing with 128 bits of plaintext (Pj). Where 1<= j <= m. If length P>128, then padding and split into blocks (m), each block 128 bits. Then, apply the proposed function fj (m) times. The outcomes of the Encryption phase (Eoutput1, Eoutput2, Eoutput3) go to the next phase (finalization) if j>m.

Algorithm 4: Encryption phase (EncAlgorithm4)

```
Input: P[1…m], Aoutput1, Aoutput2, Aoutput3
Output: C[1…m], Eoutput1[1…n], Eoutput2[1…n],
Eoutput3[1…n]
Begin
Eoutput1[1]= Aoutput1; Eoutput2[1]= Aoutput2;
Eoutput3[1]= Aoutput3;
For j=1 to m
Input1[j]= P[j] ⊕ Eoutput1[j]
C[j]= Input1[j]
Input2[j]= Eoutput2[j]
```

```
Input3[j]= Eoutput3[j]
FAlgorithm11(Input1[j], Input2[j], Input3[j], Eoutput1[j+1], Eoutput2[j+1], Eoutput3[j+1])
End for
End
```

4.4 Finalization phase on the sender side

In the fourth phase, get the three outputs of the previous step (Eoutput1, Eoutput2, Eoutput3), each with 128 bits. Get the first tag (T1) from Eoutput1. Then, apply the proposed function f one time. The outcome of the finalization phase is the second tag (T2) by XORing the function outputs (Foutput1, Foutput2, and Foutput3).

Algorithm 5: Finalization encryption phase (FinEnAlgorithm5)

```
Input: Eoutput1, Eoutput2, Eoutput3
Output: T1, T2
Begin
Input1= Eoutput1
Input2= Eoutput2
Input3= Eoutput3
T1= Input1
FAlgorithm11(Input1, Input2, Input3, Foutput1, Foutput2, Foutput3)
T2= Foutput1⊕ Foutput2 ⊕ Foutput3
End for
End
```

4.5 Decryption phase on the receiver side

On the receiver side, use the same IV/Key with the first and second phases (Initialization and Association data). The decryption phase gets the three outputs of the previous step (Aoutput1, Aoutput2, Aoutput3), each with 128 bits. Update Aoutput1 by XORing with 128 bits of ciphertext (Cj). Where 1<= j <= m. Then, apply the proposed function fj (m) times. The outcomes of the Decryption phase (Doutput1, Doutput2, Doutput3) go to the next phase (finalization) if j>m.

Algorithm 6: Decryption phase (DecAlgorithm6)

```
Input: C[1…m], Aoutput1, Aoutput2, Aoutput3
Output: P[1…m], Doutput1[1…n], Doutput2[1…n],Doutput3[1…n]
Begin
Doutput1[1]= Aoutput1; Doutput2[1]= Aoutput2; Doutput3[1]= Aoutput3;
For j=1 to m
Input1[j]= C[j] ⊕ Doutput1[j]
Input2[j]= Doutput2[j]
Input3[j]= Doutput3[j]
FAlgorithm11(Input1[j], Input2[j], Input3[j], Doutput1[j+1], Doutput2[j+1], Doutput3[j+1])
End for
End
```

4.6 Finalization phase on the receiver side

The fourth phase is the same as finalization on the sender side but different in the inputs. It gets the three outputs of the previous step (Doutput1, Doutput2, Doutput3), each with 128 bits. Get the first tag (T1) from Doutput1. Then, apply the proposed function f one time. The outcome of the finalization phase on the receiver side is the second tag (T2) by XORing the function outputs (Foutput1, Foutput2, and Foutput3).

Algorithm7: Finalization decryption phase (FinDeAlgorithm7)

```
Input: Doutput1, Doutput2, Doutput3
Output: T1*, T2*
Begin
Input1= Doutput1
Input2= Doutput2
Input3= Doutput3
T1*= Input1
FAlgorithm11(Input1, Input2, Input3, Foutput1, Foutput2, Foutput3)
T2*= Foutput1⊕ Foutput2 ⊕ Foutput3
```

4.7 Finalization phase on the receiver side

The fourth phase is the same as finalization on the sender side but different in the inputs. It gets the three outputs of the previous step (Doutput1, Doutput2, Doutput3), each with 128 bits. Get the first tag (T1) from Doutput1. Then, apply the proposed function f one time. The outcome of the finalization phase on the receiver side is the second tag (T2) by XORing the function outputs (Foutput1, Foutput2, and Foutput3).

Algorithm7: Finalization decryption phase (FinDeAlgorithm7)

```
Input: Doutput1, Doutput2, Doutput3
Output: T1*, T2*
Begin
Input1= Doutput1
Input2= Doutput2
Input3= Doutput3
T1*= Input1
```

Received: October 11, 2022. Revised: December 14, 2022.

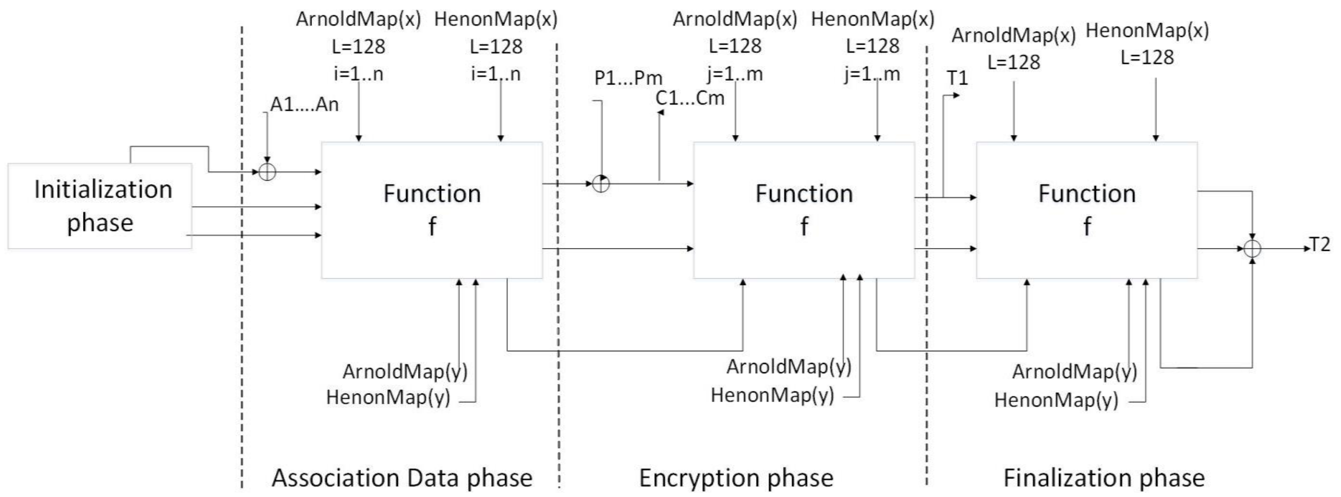

Figure. 1 Proposed an authentication encryption system based on a sponge structure

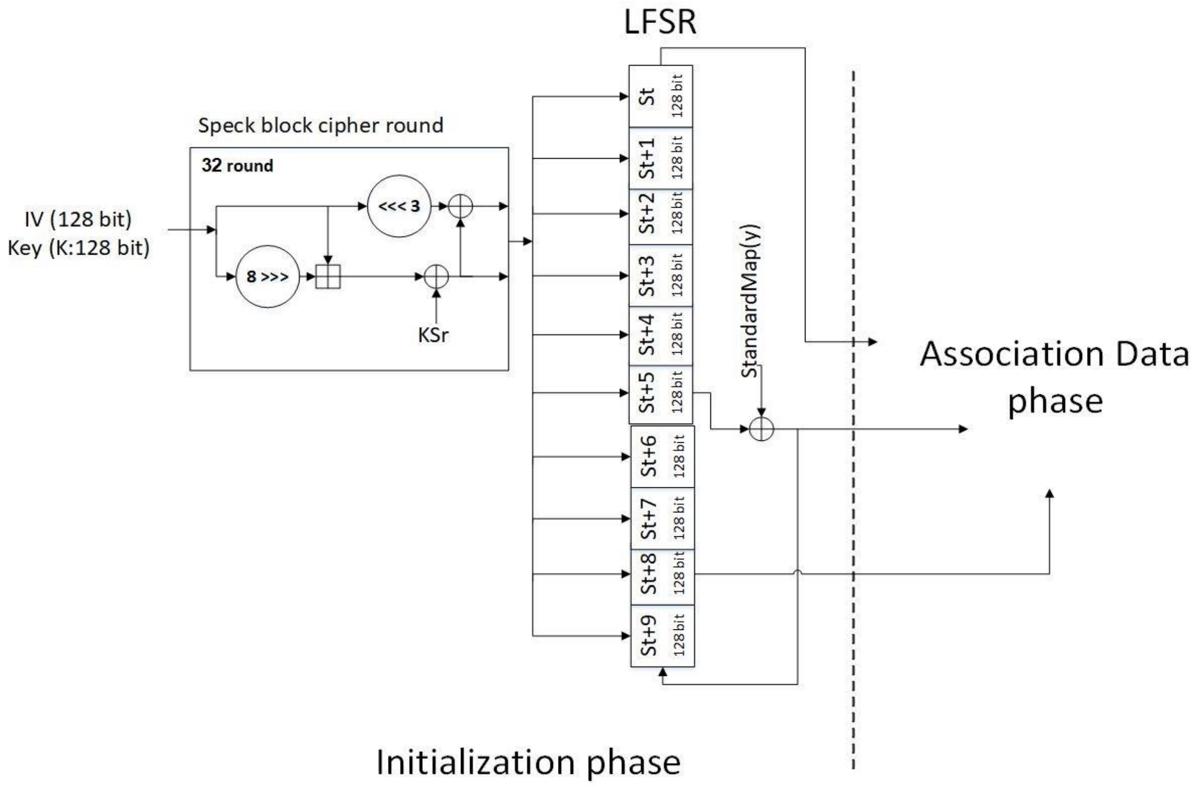

Figure. 2 The proposed initialization phase

```
FAlgorithm11(Input1, Input2, Input3, Foutput1, Foutput2, Foutput3)
T2*= Foutput1⊕ Foutput2 ⊕ Foutput3
End for
End
```

Received: October 11, 2022. Revised: December 14, 2022.

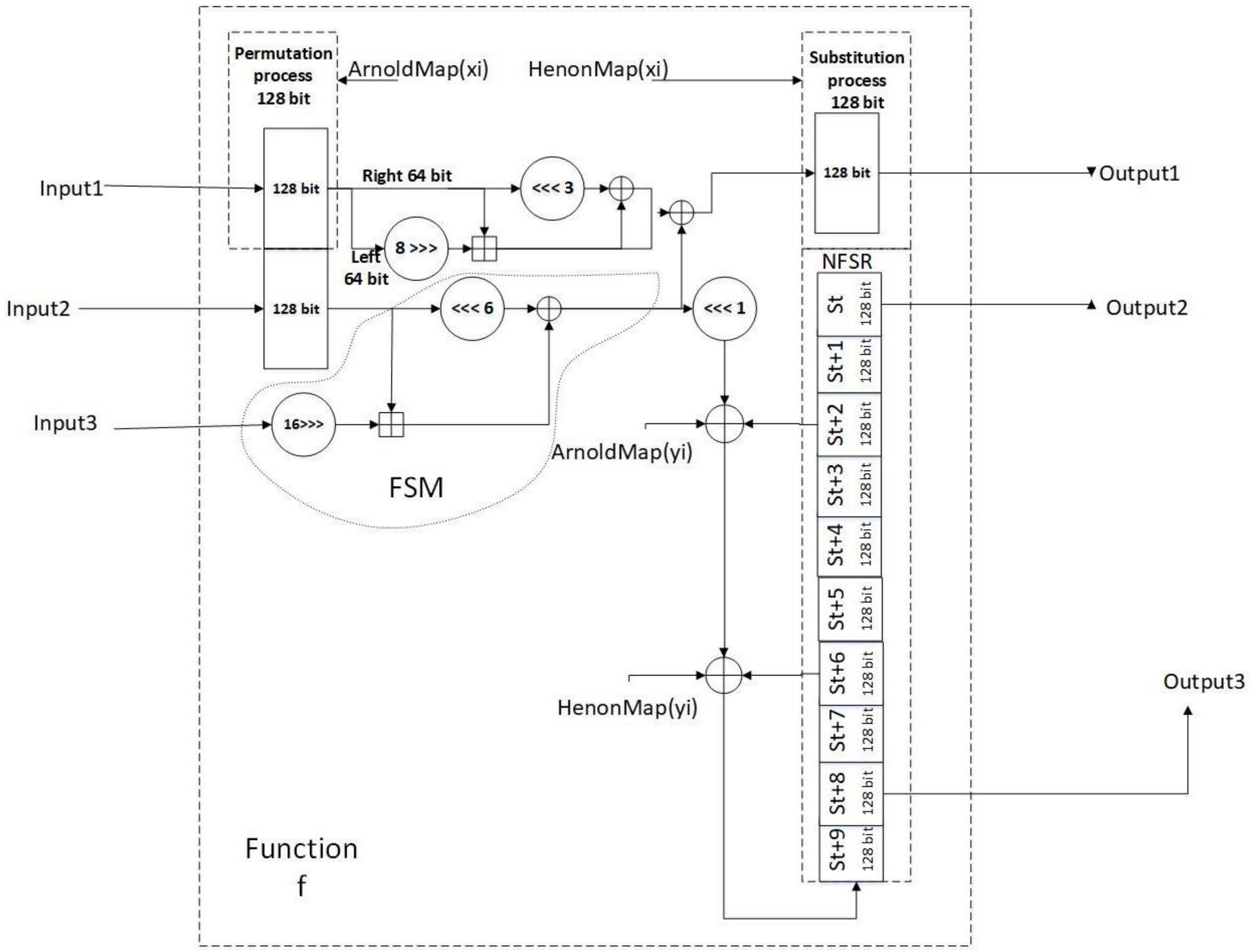

Figure. 3 The generally proposed function f

4.8 Function f

The function f has three inputs (Input1, Input2, Input3) in parallel lines. Arnold chaotic map (the output (x) of Eq. (1)) permutes the first input (Input1) and splits the result into two halves, each with 64 bits (left64 and right64). Left64 uses a round shift to the right (ROSR) by 8-bit. Then apply the addition modulo $2^{64} \boxplus$ and XOR processes with the right-side results. Right64 uses a round shift to the left (ROSL) by 3-bit. Then apply the XOR process with the left-side result. At last, concatenation left64 with right64 to get output1_128.

The Input2 XOR with Input3 to get output2_128. They are then applying the XOR between output2_128 and output1_128. Henon chaotic map (the output (x) of Eq. (3)) substitutes the outcome for obtaining the first output of function f (Output1).

To update NFSR, at first, apply round shift output2_128 to the left (ROSL) by 1 bit. Then XOR the outcome with St+2 and Arnold chaotic map (the output (y) of Eq. (2)). And we are continuing to use XOR, the result with St+6, and the Henon chaotic map (the output (y) of Eq. (4)) as feedback to update NFSR.

The second and third outputs of a function f are Output2=St, and Output3=St+8 as input for the following function if A or P length n > 128 or for the next phase where A or P length n =128.

Table 1. Randomness test NIST of 1st seven methods with proposed method

Test Name	Permutation [5]	**Permutation [6]**	**SPN [7]**	**Sponge [8]**	**Sponge [9]**	**SPN [10]**	**TETsponge [11]**	**The proposed algorithm**
Frequency	0.345	0.745	0.434	0.873	0.518	0.898	0.272	0.873
Frequency within a Block	0.415	0.752	0.782	0.370	0.764	0.342	0.165	0.598
Run	0.650	0.357	0.712	0.759	0.569	0.899	0.343	0.758
Longest Run of Ones in a Block	0.557	0.347	0.294	0.334	0.288	0.499	0.163	0.639
Binary Matrix Rank	0.333	0.270	0.202	0.573	0.177	0.706	0.508	0.576
Discrete Fourier Transform	0.795	0.641	0.195	0.112	0.193	0.394	0.155	0.889
Non-Overlapping Template Matching	0.761	0.919	0.293	0.954	0.953	0.119	0.793	0.843
Overlapping Template Matching	0.001	0.192	0.888	0.253	0.197	0.952	0.794	0.793
Linear Complexity	0.151	0.548	0.809	0.926	0.767	0.323	0.957	0.766
Serial	0.267	0.899	0.088	0.489	0.296	0.434	0.642	0.861
Approximate Entropy	0.509	0.473	0.348	0.432	0.169	0.272	0.232	0.571
Cumulative Sums Forward	0.735	0.148	0.883	0.943	0.291	0.959	0.198	0.844
Cumulative Sums Reverse	0.678	0.486	0.735	0.337	0.966	0.481	0.594	0.844
Random Excursions	0.806	0.707	0.826	0.463	0.880	0.858	0.968	0.931
Random Excursions Variant	0.311	0.474	0.231	0.481	0.371	0.614	0.573	0.502

4.8.1. Definition of non-linear feedback shift register (NFSR) in function f

NFSR consists of LFSR with FSM and chaotic maps.

Definition of LFSR in f function

The $F_{2^{128}}$ elements are used by the LFSR in function f. Ten 128-bit values, S1 through S10, make up the initial state at time t = 1. With the following recurrence, a new value is computed at each step:

$$\text{FSM= FSMalgorithm8;} \tag{10}$$

$$st+10 = st+2 \oplus (FSM<<<1) \oplus \text{ArnoldMap}(y) \tag{11}$$

$$st+10 = st+10 \oplus (st+6 \oplus \text{HenonMap}(y)), \tag{12}$$

where t ≥ 1 and a shift in the register. For a representation of the LFSR, see Fig. 3. The LFSR and the following feedback polynomial are related:

$$\pi(X) = X^{10} + X^{8} + X^{4} + 1 \in F_{2^{128}}\,[X] \tag{13}$$

Table 2. Randomness test NIST of 2nd eight methods with proposed method

Test Name	SALE [12]	SPN [13]	Spongent [14]	Keccak-p [15]	SPN [16]	Xoodoo [17]	SPN [19]	Ascon [20]	The proposed algorithm
Frequency	0.300	0.247	0.982	0.712	0.188	0.670	0.565	0.377	0.873
Frequency within a Block	0.379	0.217	0.178	0.671	0.104	0.485	0.557	0.234	0.598
Run	0.787	0.180	0.139	0.651	0.503	0.330	0.122	0.22	0.758
Longest Run of Ones in a Block	0.264	0.534	0.336	0.174	0.572	0.100	0.395	0.349	0.639
Binary Matrix Rank	0.733	0.511	0.644	0.718	0.109	0.677	0.680	0.189	0.576
Discrete Fourier Transform	0.564	0.681	0.880	0.856	0.369	0.618	0.293	0.245	0.889
Non-Overlapping Template Matching	0.559	0.552	0.867	0.888	0.510	0.389	0.647	0.449	0.843
Overlapping Template Matching	0.363	0.957	0.031	0.344	0.202	0.203	0.410	0.322	0.793
Linear Complexity	0.801	0.209	0.138	0.872	0.140	0.462	0.502	0.234	0.766
Serial	0.872	0.500	0.118	0.563	0.914	0.373	0.798	0.263	0.861
Approximate Entropy	0.112	0.405	0.553	0.356	0.422	0.483	0.492	0.37	0.571
Cumulative Sums Forward	0.440	0.459	0.911	0.195	0.196	0.135	0.353	0.432	0.844
Cumulative Sums Reverse	0.408	0.585	0.908	0.567	0.102	0.511	0.801	0.324	0.844
Random Excursions	0.705	0.545	0.253	0.236	0.424	0.594	0.233	0.558	0.931
Random Excursions Variant	0.191	0.710	0.660	0.346	0.382	0.573	0.111	0.308	0.502

The sequence of 128-bit $(st)_{t\geq1}$ is periodic and has a maximum period of $(2^{1280} - 1)$ since is a primitive polynomial.

4.8.2. Definition of FSM in function f

The finite state machine (FSM) is a component with two 128-bit registers, input2 and input3, to get 256 bits of memory. The FSM receives two 128 bits from the LFSR state as inputs at each step, changes the memory bits, and outputs 128 bits. Following is the FSM algorithm, where $\boxplus$ is addition modulo 2^{128}:

Algorithm 8: FSM (FSMalgorithm8)

```
Input: Input2 and Input3 each with 128 bit;
Output: FSMoutput with length 128 bit.
Begin
Left128=ROSR(Input3,16);
Left128=Left128 $\boxplus$ Input2;
Right128=ROSL(Input2,6);
FSMoutput= Right128 $\oplus$ Left128;
```

End

4.8.3. Proposed permutation and substitution algorithms in function f using chaotic maps

```
Algorithm 9: Permutation (PboxAlgorithm9)
Input: Input1 with length 128 bit.
Output: Input1P with length 128 bit.
Begin
1. Read Input1 as a vector with a length 128 bit.
2. Choose an initial value (x, y) as the secret key and apply the Arnold map in Eq. (1) and Eq. (2).
3. Get the vector index from 1st output (128) values of Arnold map (xi) from Eq. (1).
4. Permute the vector from step2 by ascending, which sort by the index from step 3.
5. Get the permuted text values with length 128 as P_Input1.
End
```

```
Algorithm 10: Substitution (SboxAlgorithm10)
Input: XOutput with length 128 bit.
Output: Output1 with length 128 bit.
Begin
1. choose an initial value (x, y) as a secret key and apply the Henon map in Eq. (3) and Eq. (4).
2. Get the vector of S-box with values from 1st output (128) values of Henon map (xi) from Eq. (3).
3. Read XOutput length 128 and get the value as the index of vector S-box.
4. Find the substitute value from the S-box vector.
5. Get the substituted values of the text with length 128 as Output1.
End
```

```
Algorithm 11: Function f (FAlgorithm11)
FAlgorithm11(Input1, Input2, Input3, output1, output2, output3)
Input: Input1, Input2, Input3
Output: output1, output2, output3
Begin
Input1P=PboxAlgorithm9(Input1[i], ArnoldMap(x))
InputL64=L(Input1P,64); InputR64=R(Input1P,64);
Left64=ROSR(InputL64,8);
Left64=Left64 ⊞ InputR64;
Right64=ROSL(InputR64,3);
Right64= Right64 ⊕ Left64;
Output1_128= Left64 ll Right64;
Output2_128= FSMalgorithm8(Input2, Input3);
XOutput= Output1_128 ⊕ Output2_128;
Output1= SboxAlgorithm10 (XOutput, HenonMap(x));
Output2_128=ROSL(Output2_128,1);
Output2_128= (Output2_128 ⊕ St+2) ⊕ ArnoldMap(y);
Output2_128=(Output2_128 ⊕ St+6) ⊕ HenonMap(y);
For t= 1 to 9
  S[t]=S[t+1]
  S[10]= Output2_128
Output3=st+8;
Output2=st;
End
```

Table 3. Execution time

Algorithm	**Mode/design**	**Execution Time (Sec.)**
Permutation [5]	SpongeWrap	0.620
Permutation [6]	APE	0.567
SPN [7]	CBEAM	0.508
Sponge [8]	Sp-AELM	0.477
Sponge [9]	Lsap	0.738
SPN [10]	Beetle	0.530
TETsponge [11]	Spookchain	0.478
SALE [12]	T-sponge	0.630
SPN [13]	TETsponge	0.841
Spongent [14]	CTR	0.453
Keccak-p [15]	ISAP	0.532
SPN [16]	Beetle	0.463
Xoodoo [17]	Cyclist	0.385
SPN [19]	Duplex	0.788
Ascon [20]	Duplex	0.183
The proposed algorithm	Duplex	0.37

4.9 Authentication encryption and verified decryption algorithms

At first the sender applies four phases: initialization, association data, encryption, and finalization phases to send the concatenation of ciphertext blocks and two authentication tag.
C[1]ǁC[2]ǁC[3]ǁ … ǁC[m]ǁT1ǁT2.

At receiver side, he also applies four phases: initialization, association data, decryption, and finalization phases to get two verified tags T1*ǁT2*. If T1*= T1 and T2*=T2 then get plaintext blocks P[1]ǁP[2]ǁP[3]ǁ … ǁP[m].

Algorithm 12: Authentication encryption at the sender side (SenderAgorithm12)

```
Input: Key 128 bit, IV 128 bit
Output: : C[1]ǁC[2]ǁC[3]ǁ … ǁC[m]ǁT1ǁT2
Begin
1. InitAlgorithm2(key, IV, Ioutput1, Ioutput2, Ioutput3);
2.AssoAlgorithm3(A[1…n], Ioutput1, Ioutput2, Ioutput3,Aoutput1[1…n],Aoutput2[1…n], Aoutput3[1…n]);
3.EncAlgorithm4(P[1…m], Aoutput1, Aoutput2, Aoutput3, C[1…m], Eoutput1[1…n], Eoutput2[1…n], Eoutput3[1…n]);
4.FinEnAlgorithm5(Eoutput1, Eoutput2, Eoutput3,T1, T2);
End
```

Algorithm 13: Verified decryption at the receiver side (VerifiedDeAgorithm13)

```
Input: key, IV, C[1]ǁC[2]ǁC[3]ǁ … ǁC[m]ǁT1ǁT2
Output: : P[1]ǁP[2]ǁP[3]ǁ … ǁP[m]
Begin
1.InitAlgorithm2(key, IV, Ioutput1, Ioutput2, Ioutput3);
2.AssoAlgorithm3(A[1…n], Ioutput1, Ioutput2, Ioutput3,Aoutput1[1…n],Aoutput2[1…n], Aoutput3[1…n]);
3.DecAlgorithm6(C[1…m], Aoutput1, Aoutput2, Aoutput3, P[1…m], Doutput1[1…n], Doutput2[1…n], Doutput3[1…n]);
4.FinDeAlgorithm6(Doutput1, Doutput2, Doutput3,T1*, T2*);
5.If (T1*=T1) & (T2*=T2) then return P[1…m]
End
```

5. Security and functional features

A. Security properties

A.1 The initialization phase

The proposed system initializes a key using a lightweight one-round speck of speck block cipher 32 times to fill ten states of LFSR. Then, Stanard chaotic Map (y) uses in the update of NFSR for the next phase (Association data) by XORing it with the sixth state st+5. In contrast, e.g. Ascon does not have the characteristics above of using the processes in the key initialization: XOR, addition modulo 2^n, round shift to left/right, random chaotic, and update NFSR (FSM and LFSR).

A.2 The generated of Tag

In the finalization phase, our proposal generates two tags (T1 and T2). It verifies the receiver side more than e.g. ASCON, which has one tag.

Table 4. Memory space

Algorithm	**Mode/design**	**Memory space Kilo byte (kb)**
Permutation [5]	SpongeWrap	0.883
Permutation [6]	APE	0.867
SPN [7]	CBEAM	0.995
Sponge [8]	Sp-AELM	0.850
Sponge [9]	Lsap	0.778
SPN [10]	Beetle	0.805
TETsponge [11]	Spookchain	0.869
SALE [12]	T-sponge	0.723
SPN [13]	TETsponge	0.916
Spongent [14]	CTR	0.958
Keccak-p [15]	ISAP	0.932
SPN [16]	Beetle	0.895
Xoodoo [17]	Cyclist	0.778
SPN [19]	Duplex	0.854
Ascon [20]	Duplex	0.71875
The proposed algorithm	Duplex	0.738281

Received: October 11, 2022. Revised: December 14, 2022.

Table 5. The sponge algorithms with their features

Algorithm	**Mode/design**	**Parallelizable**	**online**	**Inverse-free**	**Incremental AE**	**Single-pass**	**N. misuse resist**	**Lightweight**	**BBB security**	**RUP security**
Permutation [5]	SpongeWrap	-	-	-	-	√	-	-	-	-
Permutation [6]	APE	-	-	-	-	-	√	√	-	-
SPN [7]	CBEAM	-	-	-	-	-	-	√	-	-
Sponge [8]	Sp-AELM	-	√	-	-	-	-	√	-	√
Sponge [9]	Lsap	-	-	-	-	-	-	-	-	-
SPN [10]	Beetle	-	-	√	-	√	-	√	-	-
TETsponge [11]	Spookchain	-	√	-	-	-	-	√	√	-
SALE [12]	T-sponge	-	-	-	-	-	-	-	-	-
SPN [13]	TETsponge	-	-	-	-	√	-	-	-	-
Spongent [14]	CTR	√	-	√	-	-	-	√	-	-
Keccak-p [15]	ISAP	-	-	√	-	√	-	√	-	√
SPN [16]	Beetle	-	-	√	-	√	-	√	-	-
Xoodoo [17]	Cyclist	-	-	√	-	√	-	√	-	√
SPN [19]	Duplex	-	-	√	-	-	-	-	-	-
Ascon [20]	Duplex	-	√	√	-	√	√	√	-	-
The proposed algorithm	Duplex	-	√	√	√	√	√	√	√	√

A.3 The substitution (Sbox) and permutation (Pbox)

The proposed system has a unique and dynamic Sbox and Pbox for each use of a chaotic map for each one of Sbox and Pbox, respectively. In contrast, e.g. Ascon has fixed Sbox and permutation. This weak point causes several attacks on Ascon.

A.4 Function f

The system can repeat our function f depending on the block number of Association data A(n) and plaintext P(m). the proposed function f can automatically update the NFSR (LFSR AND FSM), Sbox, Pbox, chaotic maps, and two Tags. So our function f has a flexible property with any size of A or P and automatically updates the sponge structure compared with e.g. Ascon.

A.5 Randomness NIST tests

The proposed algorithm investigates the security with good results of 100 tests of NIST randomness of the result ciphertext are shown in Tables 1 and 2.

The Table 1 shows that the p-values of the proposed algorithm with other spongy methods and shows that the proposed has good results and most of them are equal or greater than the p-values of the other spongy methods. This gives the randomness property, which makes the attacks difficult.

A.6 Execution time and memory space

The proposed algorithm investigates the security with good speed and memory level by using stream cipher and chaotic maps with sponge structure for IoT applications.

The execution time offers in a Table 3. While Table 4 shows the Memory space. The tables show that the proposed has good execution time and memory space. That means the proposed algorithm is adequate for IoT applications.

B. Functional features

Authentication encryption schemes also need the following crucial characteristics in order to be categorized for efficient or not [4]:

B.1 Parallelizability

An authentication encryption scheme is parallelizable if execute of the one block encryption does not need the encryption of other block. Decryption fits under the same definition.

B.2 Online

There are two types of encryption schemes: online and offline. An online encryption system allows the computation of the ith blocks of ciphertext after seeing the first i plaintext blocks. On the other hand, the offline outputs the tag until all message blocks have been processed. An advantage of an online system is that the recipient can carry out authentication and decryption of ciphertext block on the receiving end.

B.3 Inverse free

An authentication encryption scheme is inverse free If there is no need to execute the inverse of encryption or decryption.

Since the same code and circuit can be used for several functions, implementation costs are low.

If inverses are required, an authentication encryption system incurs additional implementation costs.

B.4 Incrementality

If, given a tag for a particular plaintext M and a previously computed ciphertext, encrypting a different plaintext M0 that differs from M only marginally is significantly quicker than encrypting M0 from scratch, then the authentication encryption method exhibits incrementality.

B.5 Single-pass

single-pass means process the plaintext just once to give confidentiality and integrity. It makes a scheme more effective than processing the data more than once.

B.6 Lightweight

This decides whether the plan is appropriate for devices with limited resources.

B.7 Release of unverified plaintext (RUP)

An authentication encryption technique should avoid disclosing the decrypted data before the verification procedure when performing decryption.

B.8 Security beyond birth bound (BBB)

The birthday bound for most authentication encryption systems, where σ is the ciphertext block length and n is the block length, is $O(\frac{\sigma^2}{2^n})$, and it guarantees security up to that point.

Table 5 shows the functional features of a proposed method compared with other previous authentication encryption schemes. The proposed method investigates all features except the parallelization feature which can be investigated in the future work.

From the Table 5 shows the proposed method has online feature since it generates sequence of ciphertext block for each plaintext block individually and concatenate with two tags.

The proposed method has inverse free feature since it has the same code to execute encryption or decryption and this given low costs.

The proposed method has incrementality feature since it exploits chaotic maps which gives different ciphertext of any significantly changes in the plaintext.

The proposed method has single pass feature since it processes the plaintext just once to give confidentiality and integrity.

The proposed method has lightweight feature since it has low memory and time. So it appropriates for devices with limited resources.

The proposed method satisfies the release of unverified plaintext (RUP) feature where using technique [44] for storing and releasing only one or only few intermediate state to process a long ciphertext with a low buffer size without storing any part of an unverified plaintext and any intermediate tag.

At last the proposed investigates the security beyond birth bound (BBB) feature where block length n=128 and it guarantees security up to $O(\frac{\sigma^2}{2^{128}})$, where σ is the length of the ciphertext block.

6. Conclusion

Lightweight authentication encryption is a system that can be applied in the internet of things (IoT) applications. But the previous works show that they have weak points with different attacks and functional features. In addition, it requires more confusion and diffusion. This paper designs an authentication encryption algorithm based on stream cipher and chaotic maps with sponge structure for IoT applications to get the confusion/diffusion properties and investigate the functional features. The proposed design has nonlinearity and randomly properties based on NFSR (LFSR with FSM and chaotic maps) to get good two authentication tags compared with e.g. Ascon. Randomness tests show that the proposed system is random and secure with investigate all the functional features except the parallelization feature which will work it in the future. And also it has good fast speed and less memory space.

Conflicts of interest

The author declares no conflict of interest.

Author contributions

The author has the following contribution: conceptualization methodology, software, validation, formal analysis, investigation, resources, data curation, writing original draft preparation, writing review and editing, visualization, funding acquisition.

Acknowledgements

We thank Mustansiriyah University for it helps to us with our research.

References

[1] M. Oudah and A. Maolood, "Lightweight Authentication Model for IoT Environments Based on Enhanced Elliptic Curve Digital Signature and Shamir Secret Share", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 5, pp. 81-90, 2022, doi: 10.22266/ijies2022.1031.08.

[2] N. Hermawan, E. Winarko, A. Ashari, and Y. Akhmad, "High Secure Initial Authentication Protocol based on EPNR Cryptosystem for Supporting Radiation Monitoring System", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 5, pp. 1-14, 2021, doi: 10.22266/ijies2021.1031.01.

[3] L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review", *Springer, Science China Information Sciences*, Vol. 63, No. 131101, pp. 1-25, 2020.

[4] M. A. Jimale, M. R. Z'aba, L. B. Kiah, M. Y. Idris, N. Jamil, M. S. Mohamad, and M. S. Rohmad, "Authenticated Encryption Schemes: A Systematic Review", *IEEE Access*, Vol. 10, pp. 14739-14766, 2022.

[5] Bertoni, G., Daemen, J., Peeters, M., and V. Assche, G. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications", In: *Proc. of Springer 18th International Conf. On Selected Areas in Cryptography (SAC 2011)*, Toronto, ON, Canada, Vol. 7118, pp. 320-377, 2011.

[6] E. Andreeva, B. Bilgin, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha, and K. Yasuda, "APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography", In: *Proc. of Springer 21st International Conf. on FSE 2014*, London, UK, March 3-5, pp. 168-186, 2014.

[7] M. Saarinen, "CBEAM: Efficient Authenticated Encryption from Feebly One-Way ϕ Functions", In: *Proc. of Springer Topics in Cryptology – CT-RSA 2014*, San Francisco, CA, USA, February 25-28, pp. 251-269, 2014.

[8] M. Agrawal, D. Chang, and S. Sanadhya, "A New Authenticated Encryption Technique for Handling Long Ciphertexts in Memory Constrained Devices", *International Journal of Applied Cryptography*, Vol. 3, No. 3, pp. 236-261, 2017.

[9] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, and T. Unterluggauer. "ISAP_Towards Side-Channel Secure Authenticated Encryption", *IACR Transactions on Symmetric Cryptology*, Vol. 2017, No.1, pp. 80-105, 2017.

[10] A. Chakraborti, N. Datta, M. Nandi, and K. Yasuda, "Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, Vol. 2018, No. 2, pp. 218–241, May, 2018.

[11] G. Cassiers, C. Guo, O. Pereira, T. Peters, and F. Standaert, "SpookChain: Chaining a Sponge-Based AEAD with Beyond-Birthday Security", In: *Proc. of Springer 9th International Conf. on Security, Privacy, and Applied Cryptography Engineering (SPACE 2019)*, Gandhinar, India, pp. 67-85, December 3-7, 2019.

[12] J. Degabriele, C. Janson, and P. Struck, "Sponges Resist Leakage: The Case of Authenticated Encryption", In: *Proc. of Springer 25th International Conf. on the Theory and Application of Cryptology and Information Security (Advances in Cryptology- ASIACRYPT 2019)*, Kobe, Japan, pp. 209-240, December 8-12, 2019.

[13] C. Guo, O. Pereira, T. Peters, and F. Standaert, "Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction", *IACR Transactions on Symmetric Cryptology*, Vol. 2020, No. 1, pp. 6–42, May, 2020.

[14] T. Beyne, Y. L. Chen, C. Dobraunig, and B. Mennink, "Elephant v1.1", *NIST*, Jan. 17, 2021.

[15] C. Dobraunig, "ISAP v2.0", *NIST*, Jan. 17, 2021.

[16] Z. Bao, "PHOTON-Beetle Authenticated Encryption and Hash Family", *NIST*, Jan. 17, 2021.

[17] J. Daemen, S. Hoffert, M. Peeters, G. Assche, and R. Keer, "Xoodyak, a Lightweight Cryptographic Scheme", *NIST*, Feb. 11, 2021.

[18] S. Vaudenay, "Security flaws induced by CBC padding applications to SSL, IPSEC, WTLS", In: *Proc. of Springer International Conf. on the*

Theory and Applications of Cryptographic Techniques (Advances in Cryptology-EUROCRYPT 2002). Amsterdam, The Netherlands, pp. 534-545, 2002.

[19] M. Kelly, A. Kaminsky, M. Kurdziel, M. Lukowiak, and S. Radziszowski, "Customizable sponge-based authenticated encryption using 16-bit S-boxes", In: *Proc. of IEEE International Conf. on MILCOM 2015 - IEEE Military Communications*, Tampa, FL, USA, pp. 43-48, 2015.

[20] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. Ascon v2. *NIST*. Feb. 3, 2021.

[21] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schlaffer, "Cryptanalysis of Ascon", In: *Proc. of Springer International Conf. on Topics in Cryptology – CT-RSA 2015*, San Francisco, CA, USA, pp. 371–387, April 21-23, 2015.

[22] C. Tezcan, "Truncated, Impossible, and Improbable Differential Analysis of ASCON", In: *Proc. of 2nd International Conf. on Information Systems Security and Privacy - ICISSP*, Rome, Italy, pp. 325-332, 2016.

[23] K. Ramezanpour, P. Ampadu, and W. Diehl, "A Statistical Fault Analysis Methodology for the Ascon Authenticated Cipher", In: *Proc. of IEEE International Conf. on Hardware Oriented Security and Trust (HOST)*, McLean, VA, USA, pp. 41-50, 2019.

[24] K. Ramezanpour, P. Ampadu, W. Diehl, "SCARL: Side-Channel Analysis with Reinforcement Learning on the Ascon Authenticated Cipher", *arXiv:2006.03995*, pp. 1-25, 2020.

[25] C. Tezcan, "Analysis of Ascon, DryGASCON, and Shamash Permutations", *International Journal of Information Security Science*, Vol. 9, No. 3, pp. 172-187, 2020.

[26] K. N. Ambili and J. Jose, "Reinforcing Lightweight Authenticated Encryption Schemes against Statistical Ineffective Fault Attack", *cryptoeprint:2022/041*, pp. 1-18, 2022.

[27] K. Bhargavi, C. Srinivasan, and K. Lakshmy, "Panther: A Sponge Based Lightweight Authenticated Encryption Scheme", In: *Proc. of Springer 22nd International Conf. on Cryptology in India (Progress in cryptology-INDOCRYPT 2021)*, Jaipur, India, pp. 49–70, Dec. 12-15, 2021.

[28] C. Palli, N. Jampala, and T. A. Naidu, "Sponge based lightweight authentication mechanism for RFID tags", In: *Proc. of IEEE 4th International Conf. on Security and Privacy (ISEA-ISAP)*, pp. 1-7, 2021.

[29] M. R. Z'aba, N. Jamil, M. S. Rohmad, H. A. Rani, and S. Shamsuddin, "The CiliPadi Family of Lightweight Authenticated Encryption, v1.2", *Malaysian Journal of Mathematical Sciences 15(S) December*, pp. 1–23, 2021.

[30] E. L. Mohaisen and R. S. Mohammed, "Improving Salsa20 Stream Cipher Using Random Chaotic Maps", In: *Proc. of IEEE 3rd International Conf. on Engineering Technology and its Applications (IICETA)*, pp. 1-6, 2020.

[31] A. H. Mohammed, A. K. Shibeeb, and M. H. Ahmed, "Image Cryptosystem for IoT Devices Using 2-D Zaslavsky Chaotic Map", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 2, pp. 543-553, 2022, doi: 10.22266/ijies2022.0430.48.

[32] E. Mohaisen and R. Mohammed, "Stream Cipher Based on Chaotic Maps", In: *Proc. of IEEE International Conf. of Computer and Applied Sciences (CAS)*, pp. 256 – 261, Baghdad, Iraq, 2019.

[33] E. A. Albahrani, A. A. Maryoosha, and S. H. Lafta, "Block image encryption based on modified playfair and chaotic system", *ELSEVIER, Journal of Information Security and Applications*, Vol. 51, April 2020.

[34] N. H. Ghayad and E. A. Albahrani, "A Combination of Two-Dimensional Hénon Map and Two-Dimensional Rational Map as Key Number Generator", In: *Proc. of IEEE International Conf. of Computer and Applied Sciences (CAS)*, pp. 107-112, Baghdad, Iraq, 2019.

[35] R. N. Jawad and E. A. Albahrani, "New Key Generation Algorithm based on Dynamical Chaotic Substitution Box", In: *Proc. of IEEE Al-Mansour International Conf. on New Trends in Computing, Communication, and Information Technology (NTCCIT)*, pp. 93-98, 2019.

[36] R. S. Mohammed, K. K. Jabbar, H. A. Hilal, "Image encryption under spatial domain based on modify 2D LSCM chaotic map via dynamic substitution-permutation network", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol. 11, No. 4, pp. 3070-3083, 2021.

[37] E. Mohaisen and R. Mohammed, "Improving Salsa20 Stream Cipher Using Random Chaotic Maps", In: *Proc. of IEEE 3rd International Conf. on Engineering Technology and its Applications (IICETA)*, Najaf, Iraq, pp. 1-6, 2021.

[38] R. N. Jawad and E. A. Albahrani, "A New Cipher

Based on Feistel Structure and Chaotic Maps", *Baghdad Science Journal*, Vol. 16, No. 1, pp. 270-280, 2019.

[39] A. Bossert, S. Cooper, and A. Wiesmaier, "A comparison of block ciphers SIMON, SPECK, and KATAN", *Sematic Scholar,* pp.1-17, 2016.