Reinforcing Lightweight Authenticated Encryption Schemes against Statistical Ineffective Fault Attack

AMBILI K N*, JIMMY JOSE†

Department of Computer Science and Engineering,
National Institute of Technology Calicut, India

Received 31 October 2021; In final form –

The increasing use of resource limited devices with less memory, less computing resource and less power supply, motivates the adoption of lightweight cryptography to provide security solution. ASCON is a finalist and GIMLI is a round 2 candidate of NIST lightweight cryptography competition. ASCON is a sponge function based authenticated encryption (AE) scheme suitable for high performance applications. It is suitable for use in environments like Internet of Things (IoT) where large number of very constrained devices communicate with high-end servers. The drawback is that fault analyses like Statistical Ineffective fault attack (SIFA) and Sub-Set Fault Analysis (SSFA) are possible. GIMLI is also a sponge function based AE scheme which is susceptible to SIFA. In this work, we modify ASCON 128a and GIMLI exploiting the pseudo-random properties of Cellular Automata (CA) to prevent these attacks. We analyse and show that these attacks are inapplicable in the reinforced cipher.

Key words: ASCON, GIMLI, Fault analysis, Pseudorandom, Cryptography, Cellular Automata, Authenticated Encryption, SSFA, SIFA

* email: ambili_p180002cs@nitc.ac.in
† email: jimmy@nitc.ac.in

1 INTRODUCTION

The changing landscape of electronic devices and technologies involved has created new demand for security of devices and software. The conventional methods of providing privacy and authentication separately are not sufficient to address their simultaneous need.

AE algorithms are used to achieve confidentiality and authenticity simultaneously. If a passive adversary cannot determine the content of ciphertext, the AE scheme is said to have privacy. If an active adversary cannot successfully forge a ciphertext C, a nonce N and a tag t and mislead the receiver, authenticity is guaranteed. AE schemes are the modes of operation. They are algorithms built on top of primitives which are proven to be secure. These algorithms are not theoretically proven secure like the block ciphers. The security of AE modes depend on the security of the underlying primitives. We briefly outline the evolution of AE schemes before describing the algorithms under consideration.

The most primitive method of AE involves independent use of algorithms for encryption and authentication. This is called the generic composition. The strength depends on the strength of the underlying encryption scheme. Authenticated Encryption schemes provide both privacy and integrity of the transmitted messages. Often, messages have associated data with them such as the receiver's IP address. Here, it is prudent to use Authenticated Encryption with Associated Data(AEAD) [17] schemes. There are three types of AEAD processes which are Encrypt-and-MAC, Encrypt- then-MAC and MAC-then-Encrypt.

There are several methods in literature which improved the generic composition. Single-pass combined mode is one such method. Integrity Aware Parallelizable Mode (IAPM) discussed in [12] developed by Jutla at IBM in 2000 is the first such approach. Offset Codebook Mode (OCB) described in [13] was later introduced by Rogaway et. al. It is designed to be fully parallelizable and has a host of other improvements. IAPM and OCB are patented. Hence, two-pass combined modes were introduced so that the new modes are patent-free. Counter with CBC-MAC (CCM) [18], Encrypt-then-Authenticate-then-Translate (EAX) [2] and Carter-Wegman-Counter (CWC) [9] modes belong to this category.

Robust AE designs based on stream ciphers, block ciphers or sponge functions have been proposed in the last decade. National Institute of Standards and Technology (NIST) selects algorithms as part of lightweight cryptography project described in [15]. The round 2 candidates of NIST lightweight

cryptography competition include ASCON and GIMLI. ASCON is also a finalist of the competition. However, these have been shown to be susceptible to fault attacks. We consider three specific cases and propose methods to invigorate the AE schemes using Cellular Automata (CA).

ASCON [6] and GIMLI [15] are an AEAD scheme which follows Encrypt-then-MAC. ASCON is vulnerable to fault attack by double fault injection, wherein two faults are injected at two different locations. We propose enhnacements to ASCON and GIMLI using CA to prevent fault attacks and provide mathematical validation for the same.

The rest of the paper is organized as follows: Section 2 provides a brief description of algorithms. Section 3 describes the fault attacks mounted on them. Section 4 describes CA and specific features of programmable cellular automata (PCA). Our enhanced AE schemes are described in section 5. The security analysis of enhanced algorithms are provided in section 6. The strength of the algorithm to resist fault attacks is elaborated. Section 7 concludes the paper.

2 PRELIMINARIES

The NIST lightweight cryptography competition has several candidates. Here, we consider ASCON and GIMLI described in [15]. These are found susceptible to fault attacks recently. We describe them briefly with significance to their vulnerabilities.

2.1 ASCON

ASCON is a lightweight authenticated encryption cipher. ASCON [6] is based on duplex sponge modes operation which takes in data and squeezes out the modified data. This is done using core permutations p^a which is used in initialization and finalization and p^b which is used in processing of associated data and plaintext. Here, a and b represent the number of rounds for which the permutation p is run. ASCON is a light-weight 320-bit state sponge cipher with state matrix S. The initial state is given by concatenating the Initial Vector (IV) of 64 bits with key and nonce of 128 bits each. It is then processed in two parts known as rate and capacity. The first part is known as rate of r bits. ASCON uses rate of 128 bits. The second part is of length $c = 320 - r$ bits and is known as capacity. We focus on ASCON 128a in the current work and refer to it as ASCON. ASCON has four stages as shown in Fig. 1, namely:

- Initialization: Twelve rounds of the SPN transformation in which permutation p is applied to the initial state S followed by XORing of the key K to it.

- Processing Associated Data: The associated data are processed in blocks of length r (bitrate).

- Processing Plain-text: The plaintext is encrypted blockwise to give the ciphertext.

- Finalization: The state S passes through 12 rounds of transformation p, and the key K is XORed with the last 128 bits of S to get the tag T.

The SPN permutation p consists of three sub-transformations p_c, p_s and p_l which are described below:

- p_c, the constant layer where a round specific constant is added to x_2 where S=$x_0\|x_1\|x_2\|x_3\|x_4$, $x_i, i \in 0, 1, 2, 3, 4$ are part of the block

- p_s, the substitution layer where the data is passed through 64 parallel 5-bit sliced S-boxes

- p_l, the linear diffusion layer where each x_i , where 0<=i<=4, is mixed within itself

Since ASCON is inverse-free* , the decryption can be done in the same way. The decrypted ciphertext is returned only if the tags match.

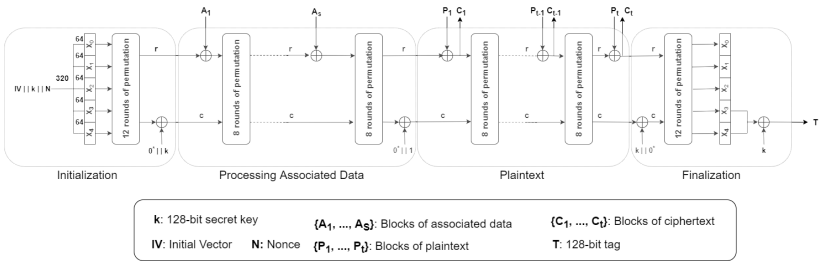

FIGURE 1: ASCON block diagram [6]

* An inverse-free cipher uses the same algorithm for encryption and decryption.

2.2 GIMLI

NIST submission of GIMLI AEAD uses a 256-bit key and 128-bit nonce and generates a 128-bit tag. GIMLI permutation applies a sequence of rounds to 384-bit state. It is a sponge based construction with 128 bit rate and 256 bit capacity. The state matrix is represented as a parallelopiped with dimensions $3 \times 4 \times 32$ which is 3×4 matrix of 32-bit words.

Algorithm 1: GIMLI Permutation [8]

Input $s = s_{i,j} \in W_{3*4}$
Output $s = s_{i,j} \in W_{3*4}$ **for** *r from 24 downto 1 inclusive* **do**
    for *j from 0 to 3 inclusive* **do**
        $x \leftarrow s_{0,j} \lll 24$
        $y \leftarrow s_{1,j} \lll 9$
        $z \leftarrow s_{2,j}$
        $s_{2,j} \leftarrow (z \leftarrow 1) \oplus ((y \wedge z) \lll 2)$
        $s_{1,j} \leftarrow x \oplus ((x \vee z) \lll 1)$
        $s_{0,j} \leftarrow y \oplus ((x \wedge y) \lll 3)$
    if $r mod 4 = 0$ **then**
        $s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{1,0}, s_{0,0}, s_{0,3}, s_{0,2}$
    else if $r mod 4 = 2$ **then**
        $s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{0,2}, s_{0,3}, s_{0,0}, s_{0,1}$
    else if $r mod 4 = 0$ **then**
        $s_{0,0} = s_{0,0} \oplus 0x9e377900 \oplus r$

GIMLI permutation is based on 384-bit state. It is represented as a three dimensional matrix of words, each of size 3×4. The rows of state matrix are named a, b and c. Columns are enumerated with subscripts 0, 1, 2 and 3. The rounds are denoted using superscript in each matrix element. The permutations are run for 24 rounds in each usage. Each round is a sequence of three operations. The algorithm for permutation is described in Algorithm 1. The first operation in each round is a nonlinear layer in which 96-bit Substitution Permutation box (SP-box described in [1]) is applied to each column. A linear diffusion layer involving a small swap and a big swap is done in every second round. In every fourth round, a constant addition is done.

For decryption, the same setup is used. The received tag is compared to tag obtained after decryption. If both the tags match, plaintext is released else empty string is output.

3 ATTACKS

A fault attack is an attack on the physical device. It leads to errors, which causes failure of the placed security systems. Fault attack is done in two steps fault injection and fault exploitation [3]. Permanent and transient fault attacks are the two types. Fault attacks which cause the device to be permanently damaged are called permanent fault attacks. Transient fault attacks are those which are not permanent and there is negligible damage to the device [7].

Statistical Ineffective Fault Attack (SIFA) was introduced as a method to attack block ciphers in 2018 [5]. It is a powerful class of attacks and is based on a fault model that is easy to achieve. The attack works based on biased distribution of an intermediate state. It does not require faulty and correct encryptions as in differential fault attack. SIFA can break traditional detection or infection based countermeasures.

SIFA is a combination of statistical fault attack (SFA) and ineffective fault attack (IFA). These differ significantly in how they achieve the fault model. The only requirement is a biased intermediate state. IFA determines the correct target partial subkey analytically whereas SFA uses a statistical approach.

IFA is the scenario in which a faulted operation always returns the same value. An attacker tries to inject ineffective fault by forcing the output to be a specific value. The subsequent steps check if the output of the faulty operation is equal to the fault free output. An ineffective fault is said to have occurred if faulted output equals fault free output.

The non-uniform distribution of intermediate values together with the corresponding faulty output is used in SFA to recover the key. In SIFA, an attacker injects a fault after computation of one function and before another one. The intermediate state thus becomes faulty. The state becomes biased. An attacker can easily achieve SFA. A partial subkey hypothesis is used to decrypt the faulty ciphertext and calculate the faulty intermediate value. This is repeated for several subkey hypotheses. The corresponding biased distribution is ranked corresponding to the intermediate values. The correct key is eventually determined.

3.1 Statistical Ineffective Fault Analysis (SIFA) on ASCON

SIFA [16] works on ASCON by injecting double faults at the bits 3 and 4 (counting starts from zero) of the output of a pair of selected S-boxes during the last round of finalization. Let X be $x_0 \| x_1 \| x_2 \| x_3 \| x_4$ where each x_i represents consecutive 64 bits of input to the linear diffusion layer L_i. The L_3 and L_4 is XORed to key K to get tag T to form equations with key K. The key

K is XORed with tag T, which is equal to the output of linear diffusion layer, as shown in Fig. 2. The output of linear diffusion layer is obtained by using a sparse matrix. The inverse of this output gives the input to the linear diffusion layer or the output of the substitution layer. The equation for bits 3 and 4 can be found for the selected pair of S-boxes, which is already known. Hence, the following equations [(1),(2) and (3)] are obtained in which the only unknown is the key [16].

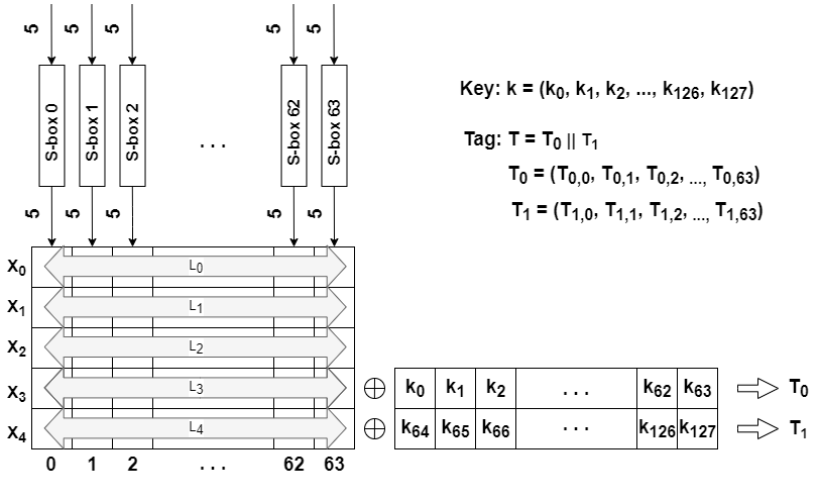

FIGURE 2: SIFA attack on ASCON

$$s_3^j = \sum_{r=0}^{63} [(T_{0,r} \oplus k_r) \odot l_{j,r}^{(3)}] \; mod 2 \tag{1}$$

$$s_4^j = \sum_{r=0}^{63} [(T_{1,r} \oplus k_{r+64}) \odot l_{j,r}^{(4)}] \; mod 2 \tag{2}$$

$$L_i^{-1} = [l_0^{(i)T}, l_1^{(i)T}, ..., l_{63}^{(i)T}]^T, \quad i = 0, 1, ..., 4 \tag{3}$$

In equations (1) and (2), s_3^j and s_4^j refers to the faulted bits 3 and 4 at the output of the selected S-boxes,
L_i^{-1} in (3) refers to the inverse of i^{th}linear diffusion layer and $l_j^{(i)T}$ is the j^{th} row of the inverse diffusion matrix coresponding to x_i. Here, x_i is the i^{th} input word of the diffusion layer. The terms k_r and k_{r+64} refers to the r^{th}

bit and $r + 64^{th}$ bit of the key K respectively. $T_{0,r}$ and $T_{1,r}$ are the r^{th} bit of first and next consecutive 64 bits of the tag T.

The entire process from double fault injection to forming equations, is repeated M times. Using the key-dividing strategy to these equations, we find the secret key K.

3.2 Sub-Set Fault Analysis (SSFA) on ASCON

SSFA [11] is done on the same path as SIFA. However, the fault is induced to the 64-bit input x_2 of the Substitution layer. It was observed that when the 3rd bit was set to zero in 10 out of 16 cases, taking the XOR of 4^{th} and 5^{th} bits results in zero. This is used here to perform the fault analysis. The analysis is done in two phases.

In the first phase, subset fault analysis is done using key partitioning. The 128-bit key is partitioned into n-bit sub-keys where n is assumed to be a power of 2. Each sub-key is a linear combination of n key bits, where coefficients of a linear combination depend on the target S-boxes used for analysis. The parity of each sub-key is used for analysis. The key hypothesis for S-box j is a set $K^j = \{P_0^{(j)}, P_1^{(j)}, ..., P_{N_{sk}-1}^{(j)}\}$ where N_{sk} is the total number of sub-keys which is $128/n$. Hence, there are $2^{N_{sk}}$ combinations for each key hypothesis. The phase 1 estimates the parity of each key hypothesis for each S-box. The flowchart for phase 1 is shown in Fig. 3. Phase 2 is not relevant to our work as the solution would make phase 1 itself ineffective.

3.3 SIFA on GIMLI

We briefly restate the SIFA attack on GIMLI based on [8]. A nonce and a hypothesis of target partial subkey k_H is used to calculate an intermediate value of state. The paper suggests reducing the number of involved key bits of the intermediate value and the number of hypothesis by attacking the early rounds of the cipher.

The attack is mounted in the initialization phase with SP-box as the target. The biased second row is attacked. It is observed that the number of hypotheses needed is approximately proportional to $2^{n_{keybits}}$ where $n_{keybits}$ denotes the number of key bits involved. It can be observed that the number of hypotheses needed grows exponentially with the number of bits of key involved. If the cipher is attacked in a later round, more number of key bits are involved and a considerably larger number of hypotheses need to be checked.

The attacks described in above subsections may be effectively thwarted with the prudent use of CA. The current work focuses on the usage of CA to

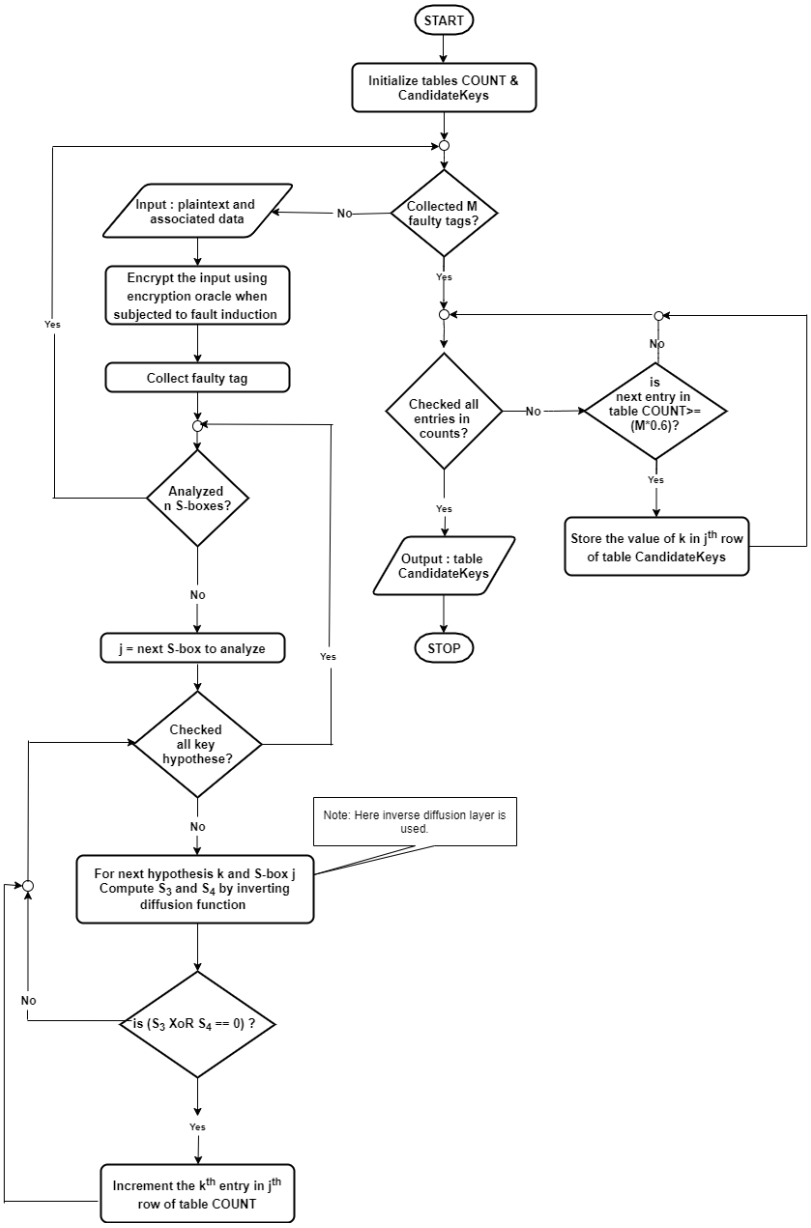

FIGURE 3: Flowchart for phase1 of SSFA

prevent fault attack. The next section describes the preliminaries of CA.

4 CELLULAR AUTOMATA

CA [19] is a lattice of cells that can take any number of values depending on its state, e.g., 2-state CA cells can take 0 or 1. Each cell value is modified in every iteration depending on a function whose parameters are the current values of the corresponding cell and its neighbor cells. In two-state three-neighbourhood CA, the neighbors are the two cells adjacent to it. The next state of a cell can be represented as the output of a function,

$$x_i(t+1) = f\{x_{i-1}(t), x_i(t), x_{i+1}(t)\} \tag{4}$$

where x_i(t) denotes the output state of the i^{th} cell at the t^{th} time step or iteration. Here f denotes the transition function of the particular cell realized with a combination logic and is known as a rule of the CA [14].
If the rules in the cells are same, then it is known as uniform CA. When the rules used in the cells are different, then it is a hybrid CA. Those CA with specific rules which result in maximum cycle length and cycle through every possible state (except 0) once before repeating the cycle of values is called maximal length CA. The rules which only involve the logical XOR are called linear or additive rules. CA can also be divided into types based on the neighbors of the extreme cells (the first and last cells). Null boundary CA refers to the extreme cell's neighbors connected to logic '0'. In the current work, null boundary maximal length CA with rules 90 and 150 are used. The rules 90 and 150 for CA are:

$$rule\,90 : q_i(t+1) = q_{i+1}(t) \oplus q_{i-1}(t) \tag{5}$$

$$rule\,150 : q_i(t+1) = q_i(t) \oplus q_{i+1}(t) \oplus q_{i-1}(t) \tag{6}$$

where q_i(t) refers to the state bit of the i^{th} cell at time t.

5 OUR PROPOSAL

We describe the modified AE schemes that use PCA 90-150 in the subsections below.

5.1 CA-based ASCON

During the Initialization and Finalization rounds of ASCON, transformation p^a (12 rounds of permutation p) is done. Since the SIFA attack takes place during the final round of Finalization, the pseudorandom number generator is used in the linear diffusion layer of p^a only. The pseudorandom generator used here is a linear hybrid cellular automata [14]. The modified algorithm for permutation is described in Algorithm 2. In the algorithm, $prng()$ function calls two CAs - CA_s and CA_r - to implement a PCA 90-150 [10]. The CA_s function uses a 6-cell, null boundary, maximal length [4] hybrid CA. This function returns the value between 1 and 63. The CA_r function is a 6-cell, maximal length, null boundary hybrid CA which selects a rule from a predefined ruleset for CA_r to use. Together this results in a simulation of PCA 90-150.

For n (number of cells) = 6, the PCA 90-150 configuration results in an expression $x(x+1)(x^4+x+1)$ of degree 6.

Algorithm 2: Permutation(byte S[])

Result: Gives the modified permutation for p_a for given S
i=0;
x0,x1,x2,x3,x4 contains the progessive eight indexes of data in S;
while $i<12$ **do**
  Addition of round constants:(no changes)
  x2 =$x2 \oplus round-constant$;
  Substitution layer:(no changes);
  .....;
  Linear Diffusion layer:
  $x0 = x0 \oplus x0 >>> prng() \oplus x0 >>> prng()$;
  $x1 = x1 \oplus x1 >>> prng() \oplus x1 >>> prng()$;
  $x2 = x2 \oplus x2 >>> prng() \oplus x2 >>> prng()$;
  $x3 = x3 \oplus x3 >>> prng() \oplus x3 >>> prng()$;
  $x4 = x4 \oplus x4 >>> prng() \oplus x4 >>> prng()$;
  i++;
end

The requirements of this pseudorandom number generator are

- The function should generate numbers from 1 to 63 randomly (number 0 is excluded because the first term of linear diffusion layer is x_i as inclusion of the value zero would make the previous and current value the same).

- The random numbers generated should be unique so as to prevent cancellation.

Considering the above requirements, a maximal length CA which generates numbers from 1 to 63 [4] are chosen. The six options for the rule set so obtained are shown in Table 1. In the first option, 0 represents rule 90 and 1 represents rule 150. The modified algorithm for permutation in ASCON is described in Algorithm 2.

option #	Rule Set
option 1	000110
option 2	101110
option 3	011010
option 4	100101
option 5	101010
option 6	100000

TABLE 1: Table showing different possible rulesets for $prng()$

The security can be further enhanced by making the random number generated by $prng()$ hard to predict by using an additional CA whose value determines which ruleset option to choose each time $prng$ is called.

5.2 CA-GIMLI

CA based $prng()$ is added to the first layer which is the non-linear layer. The modified permutation is shown in Algorithm 3 . Here, a_j, b_j, c_j denote $s_{2,j}, s_{1,j}, s_{0,j}$ respectively. The effect of $prng()$ included in the computation of a_j, b_j, c_j is cancelled during decryption. $prng()$ is included in the permutation layer. The key used in the modified scheme is generated by PCA 90-150. The additional security achieved is described in section 6.

6 SECURITY ANALYSIS

We now provide a detailed security analysis of the enhanced algorithms and show that the authenticated encryption algorithms considered are invigorated against the considered attacks with the use of PCA.

Algorithm 3: GIMLI Permutation

Input $s = s_{i,j} \in W_{3*4}$
Output $s = s_{i,j} \in W_{3*4}$ **for** *r from 24 downto 1 inclusive* **do**
    for *j from 0 to 3 inclusive* **do**
        $t_a \leftarrow s_{0,j} \lll 24$
        $t_b \leftarrow s_{1,j} \lll 9$
        $t_c \leftarrow s_{2,j}$
        $a_j \leftarrow t_c \oplus t_b \oplus prng() \oplus ((t_a \wedge t_b) \lll 3)$
        $b_j \leftarrow t_a \oplus t_b \oplus prng() \oplus ((t_a \vee t_c) \lll 1)$
        $c_j \leftarrow t_a \oplus prng() \oplus (t_c \lll 1) \oplus ((t_b \wedge t_c) \lll 3)$
    if $rmod4 = 0$ **then**
        $s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{1,0}, s_{0,0}, s_{0,3}, s_{0,2}$
    else if $rmod4 = 2$ **then**
        $s_{0,0}, s_{0,1}, s_{0,2}, s_{0,3} \leftarrow s_{0,2}, s_{0,3}, s_{0,0}, s_{0,1}$
    else if $rmod4 = 0$ **then**
        $s_{0,0} = s_{0,0} \oplus 0x9e377900 \oplus r$

6.1 ASCON

The vulnerability of ASCON 128a against SIFA and SSFA is no longer applicable to enhanced ASCON. In this section, we provide mathematical analysis of the enhanced security.

Security against SIFA

We describe the fault distribution and analysis strategies used in [16] briefly before analysing the reinforced ASCON. The fault distribution chosen for SIFA attack is the random-AND model wherein a non-uniform distribution along the diagonal of fault distribution table indicates ineffective fault. The statistical model used to determine the key from the chosen key hypothesis is the Boltzmann model [16]. It can describe the most unbiased distribution. We investigate the security of CA based ASCON against SIFA in the random-AND model under Boltzmann distribution.

The random-AND oracle model uses the equation 1, 2 and 3 to determine the fault bits s_3 and s_4. With the use of CA, the randomness in linear diffusion layer spreads. The positions of non-zero entries of the matrix L become random. This means the inverse linear diffusion matrix also becomes random. The equations for faulty bits cannot be solved reliably.

When we consider the Boltzmann model chosen to analyse the fault dis-

tribution, the key component used is Hamming weight [†] . The enhanced CA based ASCON spreads fault quickly and variation in Hamming weight will be unpredictable. This further makes it difficult to solve the Hamming weight based equations and calculate the faulted bits s_3 and s_4.

The key dividing strategy used to reduce the search space of 2^{68} to the words involved in the equation of faulty bit [16] is not possible in CA enhanced ASCON. The key dividing strategy is briefly restated here for clarity. The 128-bit length key used in the generation of tag is divided into words of length w (a power of 2). The coefficients of the combination are determined by the j-th row of the inverse diffusion matrices L_3^{-1} and L_4^{-1}. The equations for bits 3 and 4 are are rewritten as follows.

$$s_3^j = \Sigma_{s=0}^{64/w-1}(\Sigma_{r=s.w}^{(s+1).w-1} T_{0,r} \odot l_{j,r}^{(3)}) \oplus K_s^j \tag{7}$$

$$s_4^j = \Sigma_{s=0}^{64/w-1}(\Sigma_{r=s.w}^{(s+1).w-1} T_{1,r} \odot l_{j,r}^{(4)}) \oplus K_{s+64/w}^j \tag{8}$$

Thus, bits 3 and 4 at the output of S-box j can be calculated with a linear combination of bits within the key words. The key bit combinations for $s = 0, 1, ..., 128/w - 1$ are defined as

$$K_s^j = \Sigma_{r=s.w}^{(s+1).w-1} k_r \odot l_{j,r} \tag{9}$$

where $l_j = (l_j^3, l_j^4)$ has been used to simplify notation.

Hence, we can represent a key hypothesis for S-box j as

$$K^j = (K_0^j, K_1^j, .., K_{128/w-1}^j) \tag{10}$$

Every K_s^j is a linear combination of w bits of the key within the word s. Hence, we see that key bit combinations are only needed and not the entire key. There would be w binary equations for w key bits in a word. The w-bits of each keyword are calculated by solving a set of linear equations corresponding to that word.

Now, we analyse the reinforced CA based ASCON in key dividing strategy. The mixing of words in the linear diffusion layer also includes pseudorandom generators. The difficulty of calculating the mixed words thus increases. This randomisation also affects the $l_{j,r}$ used in equations involved

[†] The hamming weight of a code word is defined as the number of non-zero elements in it. The Hamming distance between two codewords is defined as the number of elements in which they differ.

in key dividing strategy. The positions of non-zero values in linear diffusion matrix becomes unpredictable. The key dividing strategy used in SIFA attack will not be feasible in CA based ASCON. This implies that brute force search on key space of size 2^{68} will be needed. SIFA thus becomes inapplicable.

Next, we consider the number of fault experiments that would be needed in CA based ASCON. The equations for SIFA (1), (2), (3) show that L_3 and L_4 are of vital importance in forming the set of linear equations. In the current design, we use a PRNG in the linear diffusion layer. The attacker should find the correct positions across which it is rotated twice. If the control bit is also randomized, $63 \times 31 \times 64$ options are to be tried out of which only one is correct. We get $63 \times 31 \times 64$ options for the two positions of L_i matrix for which the value is one. Combination is used instead of permutation because XOR is commutative, i.e., $a \oplus b = b \oplus a$. Hence, the attacker has to guess just the number generated and their order is not significant. To find L_3 and L_4, the attacker would have to guess the correct L_3 and L_4 out of $(63 \times 31 \times 64)^2$ choices for each of the M cases if control bit is not randomized. Out of the M cases, success probability is 1 out of $(63 \times 31 \times 64)^{2M}$, which is negligible. Thus, SIFA attack is rendered useless against the modified ASCON.

Security against SSFA

The flowchart for SSFA show that L_3 and L_4 are of vital importance in forming the set of linear equations. The analysis based on number of fault experiments follows on similar lines as presented in Section 6.1. Thus, SSFA becomes inapplicable on reinforced ASCON.

6.2 CA-GIMLI

There are two ways of strengthening GIMLI. CA may be used as key generator and also as PRNG in linear diffusion layer. We analyse the security achieved when CA is used in both these cases, key generation being the first case.The paper [8] suggests reducing the number of involved key bits and the number of hypotheses attacking the early rounds of the permutation.

PCA used in CA-GIMLI is a combination of two CAs namely $CA_r()$ for the ruleset and $CA_s()$ which generates the key bit. Both $CA_r()$ and $CA_s()$ are of degree 8. This implies that the each key bit generated is one among $2^8 \times 2^8$ chaoices. Consider the fault attack in which round 23 is the target and it is possible to recover two key bits as described in [8]. With CA in place, we will have $(2^8 \times 2^8)/2 = 2^{15}$ choices per bit as the chances of getting zero or one is equal. Thus, for two bits $2^{15} \times 2^{15} = 2^{30}$ accurate predictions are needed to solve the round 23 equations and recover two key bits. The fault

attack with round 22 as the target can recover 11 key bits. In the enhanced CA-GIMLI, this would become accurate prediction using $2^{15^{11}}$ choices as there are 2^{15} choices for each of the 11 key bits. The attack thus becomes impractical.

Next, we consider the security benefits of using CA based PRNG in linear diffusion layer. The PRNG generates a value between 1 and 256 which implies that there are 2^8 options. Let the plaintext space be M. Consider the attack on round 23. We will need $2^8 \times 2^7 \times M$ more experiments at least to recover two key bits. If we consider attack with round 22 as the target, 11 key bits may be recovered as shown in [8]. Among them, six can be recovered uniquely. Thus, we need to perform at least $2^6 \times 2^8 \times 2^7 \times M$ experiments in modified scheme.

In the modified CA based GIMLI, the randomness of the SP-box will be proportional to the power of the CA. Hence a larger number of hypothesis will have to be checked even if attack is mounted on first round. The number of key bits revealed will remain the same. Effectively, the attacker's job of determining the key will becomes very difficult. Hence, there is considerable benefit while using CA for key generation and also as PRNG in the linear diffusion layer.

7 CONCLUSION

The authenticated encryption schemes based on sponge construction are as secure as the underlying cryptographic primitive or the block ciphers. The proposed introduction of cellular automata based pseudorandom generator in the permutation p^a of ASCON makes the calculation of L_i^{-1} infeasible, which is needed for the SIFA and phase-1 of SSFA thereby rendering SIFA and SSFA ineffective against the modified ASCON algorithm. Also, we noticed that the proposed change prevents any attack from the trivial path produced due to XORing the key for tag calculation mentioned in [11]. In the modified ASCON, it is also observed that if the attacker already has the key, nonce, and the ciphertext, she cannot decrypt the message without the sequence used in the linear diffusion layer. The number of iterations of permutations of p^a and p^b can be reduced without reducing the security of ASCON and thereby increasing the performance of the algorithm. The resistance of GIMLI against SIFA is noteworthy. Future work include experiments on practical devices.

REFERENCES

[1] Biryukov A, (2011). Keccak specifications summary. Encyclopedia of Cryptography and Security. Springer, Boston, MA., https://doi.org/10.1007/978-1-4419-5906-5_619.

[2] M Bellare, P Rogaway, and D Wagner. (2003). A conventional authenticated-encryption mode. *IACR Eprint archive*.

[3] Olivier Benot. (2011). *Encyclopedia of Cryptography and Security*, pages 218–219. Springer US, Boston, MA.

[4] Jaydeb Bhaumik. (2015). Synthesis of all maximum length cellular automata of cell size up to 12. *arXiv preprint: https://arxiv.org/pdf/1503.04006.pdf*. Last accessed 04 September 2020.

[5] Dobraunig C, Eichlseder M, Korak T, Mangard S, Mendel F, and Primas, (2018). Sifa: Exploiting ineffective fault inductions on symmetric cryptography. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(3), 547-572.

[6] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2. submission to nist, 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf. Last accessed 04 September 2020.

[7] Christophe Giraud and Hugues Thiebeauld. (01 2004). A survey on fault attacks. *International Federation for Information Processing Digital Library; Smart Card Research and Advanced Applications VI*, 153.

[8] Michael Gruber, Matthias Probst, and Michael Tempelmeier. (2020, Last accessed 1 January 2021). Statistical ineffective fault analysis of gimli. *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), San Jose, CA, 2020, pp. 252-261, doi: 10.1109/HOST45689.2020.9300260*.

[9] Guan, Sheng-Uei, and Shu Zhang. (2004). Cwc: A high-performance conventional authenticated encryption mode. *Fast Software Encryption*.

[10] Dolores de la Guía Martínez and Alberto Peinado Domínguez. (2001). On the sequences generated by 90-150 programmable cellular automata. In *5th World Multiconference on Systemics, Cybernetics and Informatics and 7th International Conference on Information System Analysis and Synthesis (SCI/ISAS 2001, Orlando, Florida)*.

[11] Priyanka Joshi and Bodhisatwa Mazumdar. (2021). Ssfa: Subset fault analysis of ascon-128 authenticated cipher. *Microelectronics Reliability*, 123:114155.

[12] C S Jutla, (2001). A parallellizable authenticated encryption for ipsec. https://tools.ietf.org/html/draft-jutla-ietf-ipsec-esp-iapm-00.

[13] T. Krovetz, (2014). Rfc 7253: The ocb authenticated-encryption algorithm https://tools.ietf.org/html/rfc7253. https://tools.ietf.org/html/rfc7253.

[14] Sukumar Nandi, Biswajit K Kar, and P Pal Chaudhuri. (1994). Theory and applications of cellular automata in cryptography. *IEEE Transactions on computers*, 43(12):1346–1357.

[15] NIST, (2020). Lightweight cryptography csrc. https://csrc.nist.gov/projects/lightweight-cryptography/round-2-candidates.

[16] Keyvan Ramezanpour, Paul Ampadu, and William Diehl. (2019). A statistical fault analysis methodology for the ascon authenticated cipher. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 41–50. IEEE.

[17] Phillip Rogaway. (2002). Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 98–107.

[18] D. Whiting, Hifn, R. Housley, Vigil Security, N. Ferguson, and MacFergus. (2003, Last accessed 3 March 2021). Counter with cbc-mac (ccm). *Counter with CBC-MAC (CCM)*.

[19] Stephen Wolfram. (1984). Cellular automata as models of complexity. *Nature*, 311(5985):419–424.