FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 30, N° 4, December 2017, pp. 459 - 475
DOI: 10.2298/FUEE1704459S

A NOVEL ARCHITECTURE WITH SCALABLE SECURITY HAVING EXPANDABLE COMPUTATIONAL COMPLEXITY FOR STREAM CIPHERS

Prathap Siddavaatam, Reza Sedaghat

Electrical and Computer Engineering, Ryerson University, Toronto, Canada

Abstract. *Stream cipher designs are difficult to implement since they are prone to weaknesses based on usage, with properties being similar to one-time pad besides keystream is subjected to very strict requirements. Contemporary stream cipher designs are highly vulnerable to algebraic cryptanalysis based on linear algebra, in which the inputs and outputs are formulated as multivariate polynomial equations. Solving a nonlinear system of multivariate equations will reduce the complexity, which in turn yields the targeted secret information. Recently, Addition Modulo* 2^n *has been suggested over logic XOR as a mixing operator to guard against such attacks. However, it has been observed that the complexity of Modulo Addition can be drastically decreased with the appropriate formulation of polynomial equations and probabilistic conditions. A new design for Addition Modulo is proposed. The framework for the new design is characterized by user-defined expandable security for stronger encryption and does not impose changes in existing layout for any stream cipher such as SNOW 2.0, SOSEMANUK, CryptMT, Grain Family, etc. The structure of the proposed design is highly scalable, which boosts the algebraic degree and thwarts the probabilistic conditions by maintaining the original hardware complexity without changing the integrity of the Addition Modulo*2^n.

Key words: *Algebraic Attack, Modulo Addition, Algebraic Degree, Scalability, SNOW 2.0, TRIVIUM, S-Box, LFSR, NFSR, SAT solver, Stream Cipher.*

1. INTRODUCTION

In 1949, Shannon mentioned [1] the possibility of decrypting a good cryptosystem by solving a system of simultaneous equations, which describes the cryptosystem, and the equations have a large number of unknowns. One of the interpretations of this proposition [2] is to describe a cryptosystem as a system of multivariate polynomial equations and to solve this system. This has built the foundations of the cryptanalysis method commonly known as Algebraic Attack nowadays. Applications of this idea were first introduced in

Received April 6, 2017
Corresponding author: Reza Sedaghat
Electrical and Computer Engineering, Ryerson University, Toronto, Canada
(e-mail: rsedagha@ee.ryerson.ca)

[3] and [4] to break public key scheme. Later, the attack was generalized and applied on stream ciphers and block ciphers [5][6][7].

Algebraic Attack focuses on formulating multivariate polynomial equations between the inputs and outputs with low algebraic degree. In the higher echelons of security framework some methods tend to deploy a family of protocols which is designed specifically to be secure against algebraic attacks [8]. The significance of the attack is that the formulae exist with probability 1 or close to 1, unlike traditional probabilistic attacks such as differential cryptanalysis [9] and linear cryptanalysis [10]. As a result, solving such equations successfully will always yield the desire value of the targeted variable. The procedure to setup the attack typically starts with the attacker finding a set of equations that can describe the relationship between the input and the output. Each equation in the set contains an algebraic degree.

Higher degree results in higher difficulty to solve the equations. At the same time, it is very common that the number of multivariate equations is less than the number of variables. Therefore, an attacker would try to uncover ways that will lower the algebraic degree of the existing equations or new independent equations that will help describe the relationship between input and output. Moreover, it is often possible that the degree can be lowered or that new equations can be formed based on some probabilistic condition. Finally, solving the set of equations can be done through techniques such as Gaussian reduction or methods described in [11], [12], and [13].

Addition Modulo 2^n has been widely used as an elementary cryptographic module in both stream ciphers, such as CAST [14], TWOFISH [15], and MARS [16], and block ciphers, such as SOBER-t32 [17], SNOW 2.0 [18], and ZUC [19]. Typically, it is used for mixing, which combines two data sources to provide security. While the logic XOR operation is also often used for mixing, Modulo Addition offers better security against Algebraic Attack [20] because it is partly non-linear in GF(2). A linear operation in GF(2), such as XOR, can be described by an equation of algebraic degree 1. Modulo Addition is linear only at its least significant bit (LSB); therefore, it is harder for an attacker to solve using Algebraic Attack. It has been discovered that the algebraic degree of the formulae describing Modulo Addition can be reduced to quadratic [11]. At the same time, conditional properties of the Modulo Addition are also discovered to lower the algebraic degree and create new independent equations. These techniques help reduce the complexity of solving Modulo Addition tremendously. As a result, this paper aims to devise a new structure that will increase the algebraic degree when compared to the traditional Modulo Addition and increase the difficulty of using the conditional properties. At the same time, the size of the structure is user-defined and flexible, providing the users a scalable security against Algebraic Attack specifically when security is considered as a key requirement during the early stages of systems development [21][22][23][24].

The paper is structured as the following: Section 2 discusses the complexity of Algebraic Attack and describes in detail Modulo Addition under the lens of Algebraic Attack. Section 3 presents the details of the proposed design. Section 4 provides the analysis of the design and compares the design with traditional Modulo Addition. Section 5 demonstrates the application of the new design in a contemporary stream cipher example and gives the analysis of its application. Finally, Section 6 summarizes the proposed design briefly.

2. PRELIMINARIES AND TERMINOLOGY

2.1. Algebraic attack and its complexity

The steps taken typically in an Algebraic Attack can be summarized as the following:

- Formulate multivariate polynomials equations describing input-output relations with probability 1
- Explore additional independent equations supplementing the existing system of equations with probability 1 or close to 1
- Explore conditions that can lower the algebraic degree of the system of equations
- Solve the system of equations with the appropriate techniques.

Algebraic Immunity is a metric that has been developed [25] to provide a fast evaluation of the security against Algebraic Attack for a given cryptographic function. It is defined as the minimum algebraic degree in the system of equations. Moreover, the Algebraic Immunity has been deemed to be insufficient and the Describing Degree, which is the minimum algebraic degree such that an S-Box can be entirely defined by equations of that minimum degree, has been developed to provide a more thorough evaluation of security against Algebraic Attack [20]. These metrics all focus on measuring the algebraic degree of the set of equations that describes the targeted function because the degree has direct relationship to the complexity of solving the set of equations. These metrics do not attempt to address fault attacks employ that inject faults at any random location and random point of time for a stream cipher [26].

The complexity of solving Algebraic Attack has been explored in [5] and [7]. Though the complexity can vary depending on the specific method used, it can be generalized by using the estimation of the complexity of a Gaussian Elimination. When applying Algebraic Attack on a generic stream cipher, the complexity can be estimated with the following steps:

- Define R as the number of multivariate equations formed between the output bit and the states in the stream cipher
- Define N as the number of variables in the stream cipher or the number of states equivalently
- Define D as the Describing Degree of the multivariate equations
- Define T as the number of monomials of degree ≤ D and T can be calculated as:

$$T = \sum_{i=0}^{D} \binom{N}{D} \tag{1}$$

The estimated complexity can be calculated by multiplying T with the number of operations and the cycle time required for each operation. Since these parameters can be algorithm and platform dependent, T itself can be used to provide estimation. At the same time, if conditional properties are used to reduce the algebraic degree of the set of equations, the complexity increases by attaching the probability of the conditional properties happening to T.

The complexity of Algebraic Attack on a generic block cipher is slightly different. The complexity is contributed mainly from the non-linear component in a block cipher, which is typically an S-Box. A typical S-Box transforms its n-bit input variables to m-bit output variables, using a vectorial Boolean function. Using the definitions above, the number of monomials, T, can be calculated as:

$$T = \sum_{i=0}^{D} \binom{m+n}{i} \tag{2}$$

The number of equations, R, is determined by forming a matrix M of size $2^n * T$. This R is calculated as a difference given by:

$$T - (rank\ of\ Matrix) \tag{3}$$

Thereby the complexity of solving this system of equations can be estimated by:

$$\Gamma = (T/n)^{\lceil T/R \rceil} \tag{4}$$

From (4), the complexity increases when the number of monomials increases. The number of monomials increases when the algebraic degree of the set of equations increases in (1) and (2).

2.2. Addition Modulo 2^n

When viewing Modulo Addition in the eyes of Algebraic Attack, a set of equations describing the relationship between the input and output needs to be formed. This is outlined in [12] for the n-bit Modulo Addition of Z=X⊞Y, and shown in (5).

$$\begin{cases} Z_0 = X_0 + Y_0 \\ Z_1 = X_1 + Y_1 + C_0 \\ \vdots \\ Z_i = X_i + Y_i + C_{i-1} \\ \vdots \\ Z_{n-1} = X_{n-1} + Y_{n-1} + C_{n-2} \end{cases} \tag{5}$$

The variable C is used to denote the carries. The + sign in the equations denotes addition in GF(2), or simply the logic XOR operation. Each carry variable can be described by the set of equations given in (6). It can be seen from combining (5) and (6) that the Modulo Addition is partly non-linear because the LSB of the output is linear. The algebraic degree is dominated by the carry terms.

$$\begin{cases} C_0 = X_0 Y_0 \\ C_1 = X_1 Y_1 + (X_1 + Y_1)(X_0 Y_0) \\ C_2 = X_2 Y_2 + (X_2 + Y_2)(X_1 Y_1) + (X_2 + Y_2)(X_1 + Y_1)(X_0 Y_0) \\ \vdots \\ C_i = X_i Y_i + (X_i + Y_i)(X_{i-1} Y_{i-1}) + \sum_{p=0}^{i-2} X_p Y_p \prod_{q=p+1}^{i} (X_q + Y_q) \end{cases} \tag{6}$$

$$where\ 2 \leq i \leq n-2$$

From (6), the degree increases linearly with the carry terms. This is because the more significantly positioned carry terms not only depend on their corresponding input variables, which have a degree 1, but also the previous carry terms. As C_0 is generated by X_0 and Y_0, the degree of Z_1 becomes 2. Similarly, C_1 is generated by X_1 and Y_1, and the degree of Z_2 becomes 3. In fact, for an n-bit output, the algebraic degree for each output bit i is:

$$deg(i) = i + 1\ ,\ \text{where } 0 < i \leq n \tag{7}$$

As mentioned before, the complexity of solving the equations increases with the increment of algebraic degree. In [11], the author has devised a set of equations that describes Modulo Addition but limits the algebraic degree to 2. This is shown in (8).

$$\begin{gathered} Z_0 = X_0 + Y_0 \\ Z_1 = X_1 + Y_1 + X_0Y_0 \\ Z_2 = X_2 + Y_2 + X_1Y_1 + (X_1 + Y_1)(X_1 + Y_1 + Z_1) \\ \vdots \\ Z_i = X_i + Y_i + X_{i-1}Y_{i-1} + (X_{i-1} + Y_{i-1})(X_{i-1} + Y_{i-1} + Z_{i-1}) \\ \vdots \\ Z_{n-1} = X_{n-1} + Y_{n-1} + X_{n-1}Y_{n-1} + (X_{n-1} + Y_{n-1})(X_{n-1} + Y_{n-1} + Z_{n-1}) \end{gathered} \tag{8}$$

Moreover, the author is able to create in total $6n - 3$ independent equations, instead of the original n equations. This effectively reduces the complexity of Algebraic Attack on Modulo Addition even before the deployment of conditional properties.

2.3 Conditional Properties of Modulo Addition

Conditional properties have been studied in [25] and the idea can be applied to Modulo Addition in a similar fashion. The conditional properties of Modulo Addition are first explored in [26] and then expanded in [20]. The goal of using conditional equations in Modulo Addition is to lower the algebraic degree of the equations or to create more independent equations with lower degree. In general, the occurrence of these conditions is based on the manipulation of input bits and carries bits. As mentioned before, the cost of these conditions is the probability. For input bits, the probability is assumed to be uniform, or ½. For the carry bits, the probability can be generalized above in (9). The probability of a carry being 1 nears toward ½ as the number of bits increase.

$$\begin{cases} \Pr(C_i = 1) = {2^i - 1}/{2^{i+1}} \\ \Pr(C_i = 0) = 1 - \Pr(C_i = 1) \end{cases} \tag{9}$$

$$where\ 1 \leq i \leq n$$

2.3.1. Modulo Addition with no Carries

A Modulo Addition that generates no carries will have a completely linearized equation. In other words, the algebraic degree of (5) will be reduced to 1 when all carries are 0. The probability of this condition can be calculated using (9) and can be approximated to $2^{-(n-1)}$.

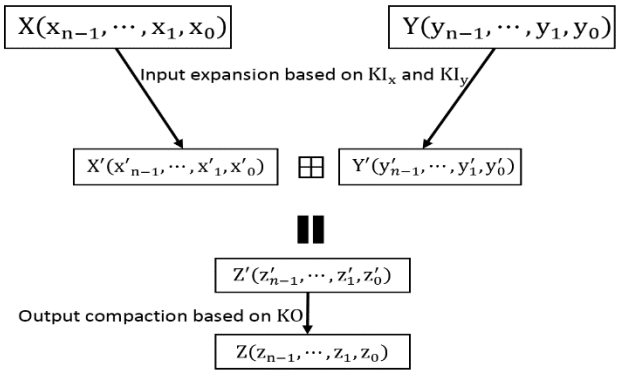

Fig. 1 Block Diagram of the New Design

2.3.2. Modulo Addition Output Characteristic

An output characteristic that can help linearize the equations is when the output bits of the addition are all 1's, or that the output is $2^{(n-1)}$. Given that the carry-in is 0, the output bits can be all 1's only when the input bits are of opposite polarity. In other words, for each pair of input bits, the two bits are either $(1,0)$ or $(0,1)$. This is also referred to as Propagate [6], and the probability of this occurring is 2^{-n}. The algebraic degree of the equations is lowered to 1 in this case.

2.3.3. Modulo Addition Input Characteristics

Two input characteristics can be utilized to linearize the equations. First, no carry is generated when one of the input is simply 0. Second, there can be no carries generated when one of the inputs is the Two's Complement of the other input. The output bits of this input pairing are always 0 in Modulo Addition. The distribution of the carry bits is as follows: There will be no carries generated from the input pairs until the first $(1,1)$ pair. Then, the subsequent input pairs will always generate a carry. This provides a controlled distribution to the carry bits. In fact, if one of the inputs is a power of 2 and the other input is the Two's Complement, then no carry will be generated. Although these conditions can reduce the algebraic degree, the probability attached to these conditions is 2^{-n}. The proposed design will not only increase the algebraic degree of the equations describing the relationship between the input and output, but also increases the difficulty of utilizing the conditional properties.

3. THE NEW MODULO ADDITION

The proposed design is a new type of cryptographic module that provides user-defined scalable security against Algebraic Attack. The components in the new Modulo Addition includes: Input Expansion, Modulo Addition, and Output Compaction. A block diagram is shown in Figure 1 to contrast the new design and traditional Modulo Addition.

3.1. Input Expansion

The Input Expansion function, $F_{IN}()$, is a function that expands each single input bit into a 2^m-bit string based on an $n*m$-bit control string KI. We specify a user-defined parameter m that can be determined depending on the security requirement. The input control string KI typically can be generated within a cipher. The actual expansion function can be flexible; meaning, the user can substitute other expanding functions instead of the proposed one. For example, the expanding function can be an algebraic function or it can be an S-Box.

The proposed expansion function is an arithmetic relationship that is easily scalable. Also, each of its output bit is 0-1 balanced. We can define the Input Expansion function as follows: Let $X = x_{n-1}, \dots, x_1, x_0$ be an n-bit input and $KI_x = KI_{x_{n-1}}, \dots, KI_{x_1}, KI_{x_0}$ be its input control string $KI_{x_i} \in \{0,1\}^m$. Furthermore, let KI_{x_i}=$KI_{x_{i,m-1}}, KI_{x_{i,m-2}}$, ..., $KI_{x_{i,1}}, KI_{x_{i,0}}$ such that $KI_{x_{i,j}} \in \{0,1\}$, $0 \le i \le n-1$ and $0 \le j \le m-1$. Then, let X' be the expanded input such that $X' = x'_{n-1}, \dots, x'_1, x'_0$ and $x'_i \in \{0,1\}^w$, where $w = 2^m$. KI_{x_i} is treated as a decimal number in (10).

$$x'_i = F_{IN}(x_i, KI_{xi}) = \begin{cases} 2^w - 1 - 2^{KI_{xi}}; for\ x_i = 0 \\ 2^{KI_{xi}}; for\ x_i = 1 \end{cases} \quad (10)$$

Using (10), it is recommended to define the user-defined parameter $m \ge 2$ to avoid repeating values.

3.2. Addition Modulo 2^{nw}

The second component of the design takes the expanded inputs and performs Modulo Addition. Nevertheless, the number of additions now has increased from 2^n to 2^{nw} where $w = 2^m$, as the inputs have been expanded. Let $X = x_{n-1}, \dots, x_1, x_0$ be an n-bit input and $Y = y_{n-1}, \dots, y_1, y_0$ be another n-bit input. Let $X' = x'_{n-1}, \dots, x'_1, x'_0 = (x'_{(n-1)(w-1)}, \dots, x'_{(n-1)1}, x'_{(n-1)0}, \dots, x'_{1(w-1)}, \dots, x'_{11}, x'_{10}, x'_{0(w-1)}, \dots, x'_{01}, x'_{00})$ be the expanded input and $Y' = y'_{n-1}, \dots, y'_1, y'_0 = (y'_{(n-1)(w-1)}, \dots, y'_{(n-1)1}, y'_{(n-1)0}, \dots, y'_{1(w-1)}, \dots, y'_{11}, y'_{10}, y'_{0(w-1)}, \dots, y'_{01}, y'_{00})$ be the other expanded input. Then, let $Z' = z'_{n-1}, \dots, z'_1, z'_0 = (z'_{(n-1)(w-1)}, \dots, z'_{(n-1)1}, z'_{(n-1)0}, \dots, z'_{1(w-1)}, \dots, z'_{11}, z'_{10}, z'_{0(w-1)}, \dots, z'_{01}, z'_{00})$ be the sum of the Modulo Addition. Equations (8) or (5) and (6) both can be used to describe the Modulo Addition. In (11), the equations are derived from (8).

$$\begin{gathered} z'_{00} = x'_{00} + y'_{00} \\ z'_{01} = x'_{01} + y'_{01} + x'_{00}y'_{00} \\ z'_{02} = x'_{02} + y'_{02} + x'_{01}y'_{01} + (x'_{01} + y'_{01})(x'_{01} + y'_{01} + z'_{01}) \\ \vdots \\ z'_{ij} = x'_{ij} + y'_{ij} + x'_{ij-1}y'_{ij-1} + (x'_{ij-1} + y'_{ij-1})(x'_{ij-1} + y'_{ij-1} + z'_{ij-1}) \\ where\ 0 \le i \le n-1;\ 0 \le j \le w-1 \end{gathered} \quad (11)$$

3.3. Output Compaction

The final component of the new design is a function that compresses Z′ to Z, i.e. from $\{0,1\}^{nw} \rightarrow \{0,1\}^{n}$, based on n ∗ m-bit output control string KO. The function F_{out} is flexible as long as the function chosen is capable of constricting the sum. In this design, the proposed function is a 2^m: 1 MUX function. Let $KO = KO_{n-1}, \dots, KO_1, KO_0$ and $KO_I = KO_{i(m-1)}, \dots, KO_{i1}, KO_{i0}$ where $KO_i \epsilon \{0,1\}^m$. We have, $Z = Z_{n-1}, \dots, Z_1, Z_0 = F_{out}(z'_{n-1}, KO_{n-1}), \dots, F_{out}(z'_1, KO_1), F_{out}(z'_0, KO_0))$. The expression for F_{out} can be generalized in (12).

$$Z_i = F_{OUT}(z'_i, KO_i) = \sum_{p=0}^{w-1} z'_{ip} \prod_{q=0}^{m-1} (-1)^{\frac{p}{2^q}+1} KO_{pq} \tag{12}$$

Here, (-1) refers to the complement of KO_{PQ}, summation refers to logic XOR, and multiplication refers to logic AND.

An example is given below for m = 3, p and q are index variables.

$$Z_i = \sum_{p=0}^{8-1} z'_{ip} \prod_{q=0}^{3-1} (-1)^{\left\lfloor \frac{p}{2^q} \right\rfloor + 1} KO_{iq}$$

$$= z'_{i0}(-1)^1 KO_{i0}(-1)^1 KO_{i1}(-1)^1 KO_{i2} + z'_{i1}(-1)^2 KO_{i0}(-1)^1 KO_{i1}(-1)^1 KO_{i2}$$

$$+ z'_{i2}(-1)^3 KO_{i0}(-1)^2 KO_{i1}(-1)^1 KO_{i2} + z'_{i3}(-1)^4 KO_{i0}(-1)^2 KO_{i1}(-1)^1 KO_{i2}$$

$$+ z'_{i4}(-1)^5 KO_{i0}(-1)^3 KO_{i1}(-1)^2 KO_{i2} + z'_{i5}(-1)^6 KO_{i0}(-1)^3 KO_{i1}(-1)^2 KO_{i2}$$

$$+ z'_{i6}(-1)^7 KO_{i0}(-1)^4 KO_{i1}(-1)^2 KO_{i2} + z'_{i7}(-1)^8 KO_{i0}(-1)^4 KO_{i1}(-1)^2 KO_{i2}$$

$$= z'_{i0}\overline{KO_{i0}}\overline{KO_{i1}}\overline{KO_{i2}} + z'_{i1}KO_{i0}\overline{KO_{i1}}\overline{KO_{i2}} + z'_{i2}\overline{KO_{i0}}KO_{i1}\overline{KO_{i2}} + z'_{i3}KO_{i0}KO_{i1}\overline{KO_{i2}}$$

$$+ z'_{i4}\overline{KO_{i0}}\overline{KO_{i1}}KO_{i2} + z'_{i5}KO_{i0}\overline{KO_{i1}}KO_{i2} + z'_{i6}KO_{i0}KO_{i1}\overline{KO_{i2}} + z'_{i7}KO_{i0}KO_{i1}KO_{i2}$$

4. Analysis of the New Design

In this section, the proposed design is analyzed with respect to Algebraic Attack.

4.1. Probability of Carry

The probability of carry in the traditional Modulo Addition can be estimated using (9). As mentioned before, each bit of the expanded input is 0-1 balanced. The Input Expansion function can be viewed as a collection of Boolean functions such that each output bit is a $\{0,1\}^{m+1} \rightarrow \{0,1\}$ function. Each Boolean function, in this case, is 0-1 balanced because the output of the function has an equal chance of producing a 0 or 1. With this assumption, the probability of carry for the new design can be derived as below. The result shows that the formula is very similar to (9).

Let $C' = c'_{n-1}, \dots, c'_1, c'_0 = c'_{(n-1)w}, \dots, c'_{(n-1)1}, c'_{(n-1)0}, \dots, c'_w \dots, c'_{11}, c'_{10}, c'_{0w}, \dots, c'_{01}, c'_{00}$ be the carry bits generated from summing the two expanded inputs. Also, the limits are $0 \leq i \leq n-1, 1 \leq j \leq 2^m$, and $w = 2^m$.

$$\begin{cases} \Pr(c'_{01}=1)=\frac{1}{2}*\frac{1}{2}=\frac{1}{4}\ ;\ x'_{00}=y'_{00}=1 \\ \Pr(c'_{01}=0)=\frac{3}{4} \end{cases}$$

$$\begin{cases} \Pr(c'_{02}=1)=\frac{1}{2}*\frac{1}{2}*\frac{1}{4}+\frac{1}{2}*\frac{1}{2}*\frac{3}{4}+\frac{1}{2}*\frac{1}{4}*\frac{1}{2}+\frac{1}{2}*\frac{1}{4}*\frac{1}{2}=\frac{3}{8}\ ; \begin{cases} x'_{01}=y'_{01}=c'_{01}=1 \\ x'_{01}=y'_{01}=1\ ,c'_{01}=0 \\ x_{01}=c'_{01}=1\ ,y'_{01}=0 \\ y'_{01}=c'_{01}=1\ ,x'_{01}=0 \end{cases} \\ \Pr(c'_{02}=0)=1-\Pr(c'_{02}=1)=\frac{5}{8} \end{cases}$$

$$\begin{cases} \Pr(c'_{ij}=1)=2^{i*w+j}-1/_{2^{i*w+j+1}} \\ \Pr(c'_{ij}=0)=1-\Pr(c'_{ij}=1) \end{cases} \quad 0\le i\le n-1\ ;\ 1\le j\le 2^m\ ;\ w=2^m \tag{13}$$

4.2. New Design with No Carries

As a result of (13), the probability of carry has decreased from $2^{-(n-1)}$ to $2^{-(wn-1)}$ for the same n-bit input pair. This helps increase the difficulty of an attacker to create a scenario without any carry, as discussed in Section 2.

4.3. Modulo Addition Output Characteristics in the New Design

In a traditional Modulo Addition, the output bits can be used directly to derive potential carries and input pairings. In the new design, the Output Compaction function is lossy; thus, the attacker can only obtain n bits out of 2^{nw} bits even if the output control string KO is known. Therefore, these n bits cannot provide enough information to derive the potential carries and input pairings. However, it is still possible to have all 1's in the sum of the Modulo Addition component in the new design. This requires specific combinations of the two m-bit input control strings KI_{x_i} and KI_{y_i}. In particular, the two input control strings need to be the same while the corresponding inputs need to be a propagate pair.

As mentioned before, the probability of output being all 1's in a traditional Modulo Addition is 2^{-n}. The probability of this condition occurring in the new design is decreased to $(2^{-n})(2^{-mn})$.

4.4. Modulo Addition Input Characteristics in the New Design

Similar approach is applied to evaluate the use of input characteristics of the Modulo Addition component in the new design. First, the expanded inputs will never be all 0's when using the Input Expansion function given in (10). Therefore, this characteristic becomes invalid. Nevertheless, it is possible for the expanded inputs to be the Two's Complement of one another. By observing (10) carefully, it is discovered that there are only 3 such cases given any m and $m \geq 2$. Thus, the probability is derived to be $(3/2^{2m+2})^n$, which is significantly less than 2^{-n}.

4.5. Complexity of Solving the New Design

To evaluate the complexity of solving the new design under Algebraic Attack, the algebraic degree needs to be understood. The algebraic degree can be obtained by expressing the new design in its algebraic normal form (ANF), which describes a Boolean function using logic XOR gates [25]. The algebraic degree of each component is first studied and then the degree of the whole design is considered.

4.5.1. Algebraic Degree of Input Expansion

The algebraic degree is the monomial with the largest degree in the algebraic normal form. For the Input Expansion function, each expanded variable can be expressed in the ANF by considering itself as a Boolean function. Intuitively, the value of the expanded variable is a manipulation of the original input value based on the value of the user-defined parameter m. From the two examples given in the Table 1, it can be observed that the algebraic degree directly relates to the value of user-defined parameter m.

The Table 1 also provides a comparison summary of the algebraic degree of the input variables going into a traditional Modulo Addition and the Modulo Addition component in the new design.

Table 1 ANF of Input Expansion Function when m = 2

X_i	ANF	Algebraic Degree
x'_{i0}	$KI_{x1} \oplus KI_{x0} \oplus KI_{x1}KI_{x0} \oplus x_i$	2
x'_{i1}	$1 \oplus KI_{x0} \oplus KI_{x1}KI_{x0} \oplus x_i$	2
x'_{i2}	$1 \oplus KI_{x1} \oplus KI_{x1}KI_{x0} \oplus x_i$	2
x'_{i3}	$1 \oplus KI_{x1}KI_{x0} \oplus x_i$	2

ANF of Input Expansion function when m = 3

X_i	ANF	Algebraic Degree
x'_{i0}	$KI_{x0} \oplus KI_{x1} \oplus KI_{x1}KI_{x0} \oplus KI_{x2} \oplus KI_{x2}KI_{x0} \oplus KI_{x2}KI_{x1} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i1}	$1 \oplus KI_{x0} \oplus KI_{x1}KI_{x0} \oplus KI_{x2}KI_{x0} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i2}	$1 \oplus KI_{x1} \oplus KI_{x1}KI_{x0} \oplus KI_{x2}KI_{x1} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus xi$	3
x'_{i3}	$1 \oplus KI_{x1}KI_{x0} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i4}	$1 \oplus KI_{x2} \oplus KI_{x2}KI_{x0} \oplus KI_{x2}KI_{x1} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i5}	$1 \oplus KI_{x2}KI_{x0} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i6}	$1 \oplus KI_{x2}KI_{x1} \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3
x'_{i7}	$1 \oplus KI_{x2}KI_{x1}KI_{x0} \oplus x_i$	3

Comparison of Input Algebraic Degrees

	Input to traditional Modulo Addition	Input to Modulo Addition in the new design
Algebraic Degree	1	m

4.5.2. Algebraic Degree of Modulo Addition

The algebraic degree of the Modulo Addition component can be evaluated using (8) or (5) and (6). Equation (8) limits the algebraic degree to quadratic in the original Modulo Addition by utilizing the output variables. This is under the assumption that the output is observable. In the new design, the output variables of the Modulo Addition component may not be observable; however, it is possible to define them as additional variables so that the algebraic degree of the expression can be reduced. The drawback of this method is that the number of variables used to solve the set of equations has increased. Assuming that additional variables are used, the algebraic degree of the Modulo Addition component is at most 2m. This is because each input variable now has a degree of m and the largest degree is quadratic using (8). At this point, it can be observed that the algebraic degree has already increased by the user-defined parameter m.

In addition, it is possible to express the Modulo Addition using (5) and (6), and its algebraic degree is outlined by (7). As mentioned before, each input variable now has a degree of m. The LSB of the addition then has a degree of m and the rest of the output bits have a degree of $(i * w + j + 1)m$. Note that $0 \leq i \leq n-1, 1 \leq j \leq 2^m - 1$, and $w = 2^m$. The derivation approach is similar to what is outlined in the previous section. The degree of the carry terms increases linearly according to their bit positions; however, the degree increases in multiples of m because the expanded input variables have a degree of m. As a result, the degree of z'_{01} is generated by the multiplication of two degree-m variables x'_{00} and y'_{00}. The degree of z'_{02} can be generated by the multiplication of x'_{01}, x'_{00}, and y'_{00}, or the combination of y'_{01}, x'_{00}, and y'_{00}. The degrees are 2m and 3m. Therefore, each output variable of the Modulo Addition, z'_{ij}, has a degree of $(i * w + j + 1)m$. A comparison summary of algebraic degree is given in Table 2.

At this point, the effective increase of algebraic is m when compared to the original Modulo Addition.

4.5.3. Algebraic Degree of Output Compaction

Finally, the algebraic degree of the Output Compaction function needs to be determined. As mentioned before, the function is a 2^m: 1 logic Multiplexer (MUX) function defined by (12). This equation is itself in the algebraic normal form. Therefore, the degree can be determined simply by observing (12). The degree is m + 1 because the output of the MUX function depends on the values of all the select lines and the input. Note that the 1 comes from the assumption that the degree of the input to the MUX is 1. It needs to be substituted when the degree changes.

4.5.4. Algebraic Degree of the New Design

The algebraic degree of the new design can be determined by combining the degrees of all the components. The Table 2 provides the summary of the algebraic degree of the new design and a comparison to the traditional Modulo Addition. Note that the algebraic degree of the traditional Modulo Addition is calculated using (8).

Table 2 Comparison of Modulo Addition Algebraic Degrees

	Sum of traditional Modulo Addition	Sum of Modulo Addition in the new design
Algebraic Degree using (8)	$1 \rightarrow 2$	$m \rightarrow 2m$
Algebraic Degree using (5) and (6)	$1 \rightarrow i + 1$	$m \rightarrow (i * w + j + 1)m$

Table 3 Algebraic Degree of the New Design

	Traditional Modulo Addition	New Modulo Addition
Algebraic Degree of Input	1	m
Algebraic Degree of Addition	2	$m \rightarrow (i * w + j + 1)m$
Algebraic Degree of Output	-	m + 1
Total	$1 \rightarrow 2$	$2m \rightarrow m + (i * w + j + 1)m$

Algebraic Immunity and Describing Degree

	Traditional Modulo Addition	New Modulo Addition
Algebraic Immunity	1	2m
Describing Degree	2	$m + \left((n - 1) * 2^m + (2^m - 1) + 1\right)m$

Table 4 Complexity Comparison of the Corner Case

	Traditional Modulo Addition using (8)	New Modulo Addition Corner Case
Number of Input Variables	$2n$	$3mn + 2n$
Number of Output Variables	n	n
Number of Extra Variables	0	0
Number of Equations	$R = 6n - 3$	$R = n$
Algebraic Degree	2	$n + 1$
Condition Cost	0	2^{3mn}
Number of Monomials	$T = \sum_{i=0}^{2} \binom{3 * n}{i}$	$T = \sum_{i=0}^{n+1} \binom{3n(1+m)}{i}$
Complexity	$\Gamma = (T/2n)^{\lceil T/R \rceil}$	$\Gamma = 2^{3mn} * (T/(3mn + 2n))^{\lceil T/R \rceil}$

As described in Table 3, the Algebraic Immunity has increased by $2m$, or at least 4 for $m = 2$. The Describing Degree has increased from 2 to at least 10 for $m = 2$ and $n = 1$. In addition, it is worth noting that, an attacker can seek to lower the degree of the new design by looking for additional independent equations with lower degree or by creating extra variables.

The benefit of these methods is to be determined by the attacker. However, a corner case study is provided in the next section as a starting point.

4.5.5. Corner Case Analysis of the New Design

The corner case can be obtained by looking for a conditional property of the new design. In other words, there is a probabilistic condition that can help lower the algebraic degree. From Table 1, it can be seen that the algebraic degree of the Input Expansion function depends on the multiplication of the input control string variables. Once the variables are known, the degree falls to 1. Specifically, if the input control string has all 0's, the expanded inputs are either the same as the inputs or the complement of the inputs. Under this condition, the degree of addition becomes at least 1, which is the same as the traditional Modulo Addition. However, the degree does not scale linearly with the significance of the bit positions. In each block of expanded inputs, the expression of the summation of the expanded input variables can be reduced because many of the variables are the same. In fact, the degree of the LSB in each block of expanded inputs is $i + 1$, for $0 \leq i \leq n - 1$. The rest of the summation bits from adding each block of the expanded inputs have a degree of $i + 2$.

Furthermore, the attacker would notice that if the Output Compaction function is able to select the LSB in each block of the summation, i.e., z'_{i0}, the algebraic degree is the lowest. For this to happen, the output control string needs to be all 0's. As a result, the degree of the new design becomes $i + 1$ for Z'_i, and $0 \leq i \leq n - 1$. This is the same as the traditional Modulo Addition as shown in Table 2. Nevertheless, the cost of this condition has a probability of 2^{-3mn} as all control bits need to be 0's.

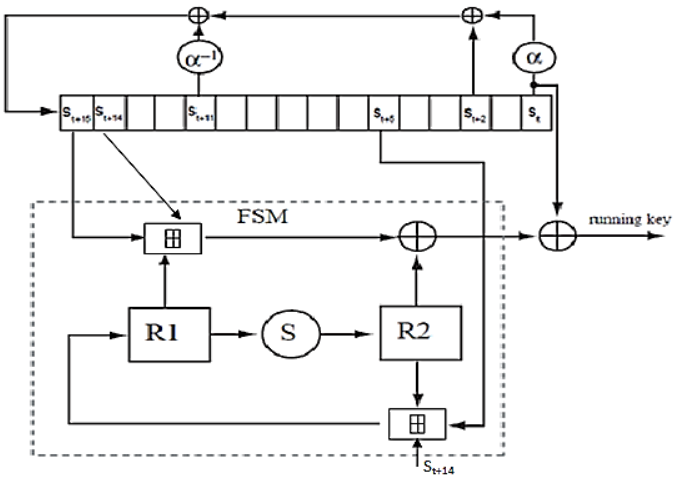

Fig. 2 New Design Schematic for SNOW 2.0

Traditional Modulo Addition and the new design can be viewed as S-Boxes and their complexity against Algebraic Attack can be approximated as S-Boxes. For the traditional Modulo Addition, the required parameters have been studied in [20]. A comparison for the same has been listed in Table 4. In the corner case, the number of monomials is still

larger because of the increased in number of variables and algebraic degree. At the same time, the complexity has increased by attaching the conditional cost.

5. Applications of New Design

In this section a contemporary stream cipher like SNOW 2.0 is used to demonstrate the application of the new design. This application explores the new design being deployed in a stream cipher that uses combiner with memory. The following example describes the new design being applied in the stream cipher, SNOW 2.0, which constitutes a LFSR with non-linear feedback and uses an output combiner function with memory [18].

5.1.1. Overview of SNOW 2.0

SNOW 2.0 uses a length 16 LFSR over $GF(2^{32})$. In other words, the LFSR has 16 elements, or states, but each state contains a 32-bit word. Let S_0 , S_1,…, S_{15} denote the states of the LFSR. The feedback function is defined as the XOR combination of S_0 multiplied by α, S_2 and S_{11} divided by α. To produce the output key stream, a Finite State Machine (FSM) is used in conjunction with the LFSR. The FSM contains two 32-bit registers R1 and R2. The value of R2 is determined by feeding the value of R1 through a set of AES S-Boxes and the AES Mix Column function. The value of R1 is determined by performing Addition Modulo 232 between R2 and S_5. Finally, the output combiner function is defined as first performing Addition Modulo 2^{32} between R1 and S_{15}, second XORing the result with R2, and finally XORing the result of the former with S_0. Consequently, SNOW 2.0 operates in two modes: Initialization and Key Stream Generation; it uses a 128-bit secret key and a 128-bit initialization vector in Initialization Mode.

5.1.2. Application of the New Design

The new design is used to replace the two Modulo Additions and the user-defined parameter m is chosen to be 3. There are 288 extra bits required to supply the input and output control strings of each addition because for each input bit of the 32-bit addition, a 3-bit control string is needed. Therefore, 576 bits in total are required for two additions. Again, many ways can be utilized to generate the extra bits. In this case, S_{14} is used to generate 288 bits and the same set of bits is used for the two insertions of the new design. The generation logic is defined as the following:

- For each bit of the first input X, there are 3 input control bits needed. They will be the 3 LSBs of the 3-bit circular-left-shifted S_{14}. For example: $KI_{x_0} = (S_{14,2}, S_{14,1}, S_{14,0})$ and $KI_{x_1} = (S_{14,31}, S_{14,30}, S_{14,29})$.
- For each bit of the second input Y, the 3 input control bits will come from the 3 LSBs of the 3-bit circular-right-shifted and inverted S_{14}. Let S'_{14} denotes the bit-wise inverted S_{14}. Then, $KI_{y_0} = (S'_{14,2}, S'_{14,1}, S'_{14,0})$ and $KI_{y_1} = (S'_{14,5}, S'_{14,4}, S'_{14,3})$.
- For the output control string, each 3 output control bits come from the 3 LSBs of the 3-bit circular-right-shifted S_{14}. For example: $KO_0 = (S_{14,2}, S_{14,1}, S_{14,0})$ and $KO_1 = (S_{14,5}, S_{14,4}, S_{14,3})$.

This setup can at least guarantee that the input control bits for the first input pair will not be all 0's simultaneously. The new SNOW 2.0 setup is shown in Table 5. The secret key

and the initialization vector are defined as given in the Table 5. This is one of the test vectors listed in [18]. At this specific timeframe, S15 = 0xCC15A50B, R1 = 0xAAB91A68, and S14 = 0x5164B6D9. The output of the new design here is 0x37F7B4F7 while the original Modulo Addition gives 0x76CEBF73. Also, the first output key stream of new SNOW 2.0 is 0x91CC022F and the original key stream is 0xC355385D.

Table 5 SNOW 2.0 Test Vectors [24]

Attribute	Hexadecimal Value
Secret Key(K)	0xAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Initialization Vector (IV)	0x00000004000000030000000200000001

5.1.3. Analysis of New SNOW 2.0

Algebraic Attack on SNOW 2.0 has been studied in [26] and [20]. Two methods have been proposed to linearize the Addition Modulo 2^{32} in the stream cipher. The first method is relatively straightforward, as the Modulo Addition can be completely linearized when there are no carries. The probability of this occurring can be estimated using (9). To be more precise, the condition is satisfied as long as each input pair does not generate a carry. The probability of this happening is $(3/4)^{31}$ because the probability of an input pair to generate no carries is (3/4). The author in [26] seeks to use this condition for both additions and for 17 consecutive cycles. The probability of this is $(3/4)^{(31*2*17)} \approx 2^{-438}$, which is close to exhaustive search 2^{-576}. In SNOW 2.0, the exhaustive search includes the search for 512 bits in the LFSR states and two 32-bit registers.

In the new SNOW 2.0, the cost of having no carries has greatly increased. As m = 3 in this application, the length of the Modulo Addition component in the new design becomes $32*2^3 = 256$. To fix the carries for one Modulo Addition, the probability is estimated to be $2^{-(31*8*17)} = 2^{-4216}$ by using (9). This is much larger than exhaustive search.

The second method sees the attacker trying to manipulate the output characteristics of the Modulo Addition to linearize the equations, as described in [20]. In particular, 9 consecutive values of the register R1 are fixed. The desired output values from the summation are $R1_1 = 0$, $R1_2 = 2^{32} - 1$, $R1_3 = 0$, $R1_4 = 0$, $R1_5 = 0$, $R1_6 = 0$, $R1_7 = 0$, $R1_8 = 0$, and $R1_9 = 0$. The value of R1 comes from summing R2 and S_5 but the value of R2 comes from feeding R1 through the ASE S-Boxes and Mix Column operation. Therefore, only S_5 needs to be fixed. Due to the nature of LFSR, 9 states need to be fixed and they are: S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, and S_{13}. The associated probability is $2^{-(32*9)} = 2^{-288}$.

With the new design applied; however, the output characteristic may not be applicable. As discussed in Section 4, the probability of fixing all outputs to be 1 in the new design is $2^{-n(m+1)}$. In this scenario, the probability has become $2^{-32(3+1)} = 2^{-128}$. In addition, the probability of fixing all outputs to be 0 in the new design is $(3 / 2^{2m+2})^n$. Again, the probability becomes $(3 / 2^{2*3+2})^{32} \approx 2^{-205}$. For a total of 9 consecutive cycles, the probability has become $2^{-205*8}*2^{-128} = 2^{-1768}$.

In essence, the adversary may want to utilize the corner case of the new design to lower the algebraic degree. However, the control string generation logic, outlined in Section 5, guarantees that the input control strings for the LSBs of the two inputs will not be simultaneously 0. Therefore, the set of equations cannot be completely linearized. A summary is provided in Table 6.

Table 6 Result Analysis of New SNOW 2.0

	SNOW 2.0	New SNOW 2.0
By Method 1: Fix Carries to 0	2^{-248}	2^{-4216}
By Method 2: Fix Consecutive Outputs	2^{-288}	2^{-1768}
Corner Case	-	NA

6. CONCLUSIONS

In this paper, a new type of Modulo Addition is proposed to defend against Algebraic Attack. It contains three components: Input Expansion, Modulo Addition, and Output Compaction. In addition, the new design utilizes an expanding and compacting structure that can be user-defined to fit into various cryptographic security architectures. The new design is capable of improving the Algebraic Immunity by 4 times, by defining the user-defined parameter m to 2, and the Describing Degree by at least 5 times, as outlined in Table 3. Although the algebraic degree can potentially fall back to the same as the original Modulo Addition, the associated cost of doing so is at least 2^{-mn}. In Section 5, the new design was applied to stream cipher like SNOW 2.0 to demonstrate the capability of the new design against Algebraic Attack. The cost of utilizing output and input characteristics of Modulo Addition has been increased to more than the exhaustive search, which is 2^{-576}, in the new SNOW 2.0. Overall, the new design can serve as a new cryptographic component that provides scalable security against Algebraic Attack.

Acknowledgement: *This work is supported by the Optimization and Algorithm Research Lab (OPRAL), Ryerson University.*

REFERENCES

[1] C. Shannon, "Communication theory for security systems," *Bell System Technical Journal 28*, 1949.
[2] C. Adams, "Designing against a class of algebraic attacks on symmetric block ciphers," Applicable Algebra in Engineering, Communications, and Computing, vol. 17, no. 1, pp. 17-27, Apr. 2004.
[3] J. Patarin, "Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms," in Advances in Cryptography – EUROCRYPT'96, Springer Berlin Heidelberg, 1996, pp. 33-48.
[4] J. Patarin, "Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88," in Advances in Cryptography – EUROCRYPT'95, Springer Berlin Heidelberg, 1995, pp. 248-261.
[5] N. Courtois and W. Meier, "Algebraic Attack on Stream Ciphers with Linear Feedback," in Advances in Cryptography – EUROCRYPT 2003, Springer Berlin Heidelberg, 2003, pp. 345-359.
[6] N. Courtois, "Algebraic Attack on Combiners with Memory and Several Outputs," in Information Security and Cryptography – ICISC 2004, Springer Berlin Heidelberg, 2004, pp. 3-20.
[7] N. Courtois and J. Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined System of Equations," in Advances in Cryptography – ASIACRYPT 2002, Springer Berlin Heidelberg, 2002, pp. 267-287.
[8] C. Adams and S. Tavares, "Designing s-boxes for ciphers resistant to differential cryptanalysis," In Proceedings of the 3rd Symposium on the State and Progress of Research in Cryptography, Feb. 1993, pp. 181-190.
[9] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptography*, vol. 4, no. 1, pp. 3-72, Jan. 1991.
[10] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," in Advances in Cryptography – EUROCRYPT'93, Springer Berlin Heidelberg, 1994, pp. 386-397.

[11] N. Courtois and J. Patarin, “About the XL Algorithm over GF(2),” in Topics in Cryptography – CT-RSA 2003, Springer Berlin Heidelberg, 2003, pp. 141-157.
[12] N. Courtois, “Higher Order Correlation Attacks, XL Algorithm and Cryptanalysis of Toyocrypt,” in Information Security and Cryptography – ICISC 2002, Springer Berlin Heidelberg, 2002, pp. 182-199.
[13] N. Courtois, A. Klimov, J. Patarin, and A. Shamir, “Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations,” in Advances in Cryptography – EUROCRYPT 2000, Springer Berlin Heidelberg, 2000, pp. 392-407.
[14] C. Adams, “Constructing Symmetric Ciphers Using the CAST Design Procedure,” in Selected Areas in Cryptography, Springer US, 1997, pp. 71-104.
[15] B. Scheier et al, The Twofish encryption algorithm: a 128-bit block cipher, New York, NY, Wiley, 1994.
[16] C. Burwick et al, “MARS – a candidate cipher for AES,” IBM Corp., Rep., 1998.
[17] P. Hawkes and G. Rose, “Primitive specification and supporting documentation for SOBER-t32 submission to NESSIE,” In the Proceedings of the first open NESSIE workshop, 2000.
[18] P. Ekdahl and T. Johansson, “A New Version of the Stream Cipher SNOW,” in Selected Area in Cryptography, Springer Berlin Heidelberg, 2003, pp. 47-61.
[19] “Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3. Document 2: ZUC Specification,” Rep. version 1.6, Jan. 2011.
[20] N. Courtois and B. Debraize, “Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0,” in Information and Communications Security, Springer Berlin Heidelberg, 2008, pp. 328-344.
[21] A. Bushager, M. Zwolinski, "Evaluating system security using Transaction Level Modelling," *Facta Universitatis, Series: Electronics and Energetics*, vol.27, issue.1, pp.137-151, 2014.
[22] A. Khanna, “An architectural design for cloud of things”, *Facta Universitatis, Series: Electronics and Energetics*, vol. 29 issue 3, pp. 357-365, 2016.
[23] A. Janjic, S. Savic, G. Janackovic, M. Stankovic and L.Velimirovic, “Multi-criteria assessment of the smart grid efficiency using the fuzzy analytic hierarchy process”, *Facta Universitatis, Series: Electronics and Energetics*, vol. 29, issue. 4, pp. 631-646, 2016.
[24] M. A. Dimitrijević, M. Andrejević-Stošović, J. Milojković, V. Litovski, " Implementation Of Artificial Neural Networks Based AI Concepts To The Smart Grid ", *Facta Universitatis, Series: Electronics and Energetics*, vol.27, issue.3, pp.411-424, 2014.
[25] W. Meier, E. Pasalic, and C. Carlet, “Algebraic Attacks and Decomposition of Boolean Functions,” in Advances in Cryptography – EUROCRYPT 2004, Springer Berlin Heidelberg, 2004, pp. 474-491.
[26] S. Sarkar, S. Banik and S. Maitra, "Differential Fault Attack against Grain Family with Very Few Faults and Minimal Assumptions," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1647-1657, June 2015.