OIC-CERT Journal of Cyber Security
Volume 3, Issue 1 (April 2021)
65 - 74

New Vulnerabilities upon Grain v0 Boolean Function through Fault Injection Analysis

Wan Zariman Omar@Othman[1,2], Muhammad Rezal Kamel Ariffin[2], Suhairi Mohd. Jawi[1], and Zahari Mahad[2]
[1,] CyberSecurity Malaysia, Cyberjaya, Malaysia
[2]Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia (UPM) Serdang, Malaysia
wanzariman@cybersecurity.my

ARTICLE INFO

Article History
Received 16 May 2020
Received in revised form 26 Mar 2021
Accepted 30 Mar 2021

Keywords:
Vulnerabilities;
Boolean function;
Fault Injection Analysis (FIA);
Stream Cipher;
Annihilator

ABSTRACT

Algebraic attacks on stream cipher are very important in cryptography as well as in cryptanalysis. Generally, increasing degree of the equation will make an algebraic attack to the equation hardest. In conducting this analysis, we aim to decrease the degree of the targeted Boolean equation by constructing low degree annihilator equation(s). We adopt the Fault Injection Analysis (FIA) methodology to achieve our objectives. In this study, we found annihilator(s) through FIA (inject with value of one (1)) on Boolean function of selected stream ciphers. With the new injected Boolean functions developed, we proceed to utilize Hao's method to find new annihilator(s). Then we established new annihilator(s) of Grain v0's Boolean function. As a result, these newly identified annihilator(s) successfully reduce the complexity of the published Boolean function to guess the initial secret key. It also provides much needed information on the security and vulnerability of these selected stream cipher with respect to FIA.

I. INTRODUCTION

The objective of security is to protect against those who may harm intentionally or unintentionally. Security can be seen in many organizations, but this research prioritizes communication and information security. Communication security protects technology, media, and content. Meanwhile, information security protects the confidentiality, integrity, and availability. To ensure our information is secure, cryptology is one aspect to consider. As we all know, cryptology is a science that has two part:

1) Cryptography.
2) Cryptanalysis.

Cryptography originated came the Greek words that **kripto** and **graphia** which means *hidden* and *writing*. This technology of securing

ISSN 2636-9680
eISSN 2682-9266

messages began since early civilization when human started to communicate and the need to keep their communication secret . The fundamental and classical task of this science is to provide confidentiality by encryption methods where both the encryption and decryption process used a secret key that was initially agreed by both parties. [1]

Cryptography has two types: asymmetric cryptography and symmetric cryptography. For symmetric cryptography, only one key will be used to encrypt and decrypt the data. Meanwhile for asymmetric cryptography, two different keys will be used to encrypt and decrypt. Asymmetric cryptography is also known as public key cryptography, it uses a pair of keys known as public and private keys to encrypt and decrypt data:

1) Public key: a key that can be shared with everyone and it is the key pair of the private key.
2) Private key: a key that must be kept secret by the owner.

For secret information transmission and storage, usage and implementation of symmetric key is very important. Both parties, the sender and receiver, share the same secret key. The sender and the receiver share the same secret key. To obtain the ciphertext, the sender must encrypt the message (plaintext) with a cipher and key. Ciphertext is usually transmitted over an insecure channel. The recipient must decode the ciphertext to get the original message with the same secret key. An attacker may decrypt the ciphertext, so a strong algorithm and strong key is highly recommended and should be used for encryption to ensure that the attacker does not have any information leaked. Rueppel points out the variations as in [2].

1) Block cipher: Operate with a fixed transformation on large block of plain text data.
2) Stream cipher: Operate with a time-varying transformation on individual plain text digits.

A. Attacks in Cryptography

In building a cryptosystem, a developer usually mathematician/cryptologist will build his or her best cryptographic algorithm meanwhile a cryptanalyst (also mathematician) will take opportunity to tackle the method of breaking the cryptosystem. All single analysis of and attack on each cryptosystem is very important because it will used to be a criterion to strengthen that particular cryptosystem. By [3], attackers in cryptography can be divided into two types:

1) Passive attacker
2) Active attacker and there are six (6) types of active attacks:
   a) Chosen-plaintext attack
   b) Chosen-ciphertext attack
   c) Ciphertext only attack
   d) Known plaintext attack
   e) Adaptive chosen-plaintext attack
   f) Adaptive chosen-ciphertext attack

B. Types of Stream Ciphers Attacks

Before we do an attack on or analysis of any stream cipher algorithms, it is very important to learn and understand all the possible attacks in stream cipher and the main purpose is to recover or discovering the key used in the process of encryption and decryption. By [4], there are ten (10) types of attacks in stream ciphers and each attack has their own method to recover or discovering the keys that were used. So from all ten (10) attacks, we are more focusing in Algebraic Attack and Fault Attack.

C. Boolean Functions in Stream Cipher

This subsection provides introduction to Boolean functions [5].

Definition 1. (Boolean function). A Boolean function on n may be viewed as a mapping from $\{0,1\}^n$ into $\{0,1\}$.

A Boolean function $f(x_1,...,x_n)$ is also can be written as the output of its truth table f.

Definition 2. (Algebraic normal form of Boolean function - ANF). Every Boolean function f can be expressed as a multivariate polynomial over F_2. This polynomial is known as algebraic normal form of the Boolean function f.

Eq. 1 below showed the definition about Algebraic normal form of Boolean function.

Eq. 1. Algebraic normal form.

$$f(x_1,\ldots,x_n) = a_0 \oplus \bigoplus_{1\leq i\leq n} a_1 x_i \oplus \bigoplus_{1\leq i<j\leq n} \oplus\cdots\oplus a_{12......n} x_1 x_2 \ldots x_n$$

Definition 3. (Degree of Boolean function) Degree of a Boolean function f is defined as $\deg(f)$ = number of variables in the highest order product term in the algebraic normal form of f. Functions of degree at most one are called affine function. An affine function with constant term equal to zero is called a linear function.

Definition 4. (Annihilator of a Boolean function) A non-zero Boolean function g of n variables is said to be an annihilator of a Boolean function

$$f \Leftrightarrow g(X)\cdot f(X) = 0, \forall X \in \{0,1\}^n$$

D. Annihilator

As we mention in Definition 4, we let $g \in B_n$ an annihilator of f function if $f \cdot g = 0$ or all $x \in \{0,1\}^n$. By [6], the existence of low-degree equations can be divided into three scenarios:

1) Scenario S3a: Assume that there exists a function g of low degree such that the product function is of low degree, as example $f \cdot g = h$, where h is a non-zero function of low degree.
2) Scenario S3b: Assume there exists a function g of low degree such that $f \cdot g = 0$.
3) Scenario S3c: Assume there exists a function g of high degree such that $f \cdot g = h$ where h is non-zero and low degree.

But in 2004, [7] has reduced and improved method to find existence of low-degree equation to only one scenario. But via [8], we can effectively calculate

ISSN 2636-9680
eISSN 2682-9266

all low-degree annihilators of both f and $1+f$. Therefore, we choose this method together with FIA to obtained annihilator(s).

II. RELATED WORK

This paper focused on the cryptanalysis of stream cipher algorithm via their Boolean function. From previous work as [8] [9], [10], there was few work or cryptanalysis on Grain family. So, from all this previous work, we narrow the research scope using fault injection attack as we refer to [11], [12], [13], [14] and [15].

III. METHODOLOGY

This section explains the research design used for conducting this research. The first step is collecting and understanding previous works. Secondly in Step 2, we will define each default general Boolean function of each selected stream cipher including identifying how many monomials (n) degree d of Boolean function.

A. Boolean Function

Given Boolean function

$$h(x) = x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4 \qquad (1)$$

where $n = 5$ variables and $d = 3$ (degree of Boolean function);

B. Fault Injection

This subsection explains how to inject fault value on Boolean function and generate a set of injected Boolean function. In this paper, we will inject (replace) value of one (1) to each active coefficient in each Boolean function. Replace each active coefficient of Boolean function *h(x)*, starting with $x_0 + x_1 +, \dots, + x_2x_3x_4 = 1$;

Let Boolean function$h(x) = x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$.

We define the following notation from Boolean function f $(x_0, x_1, x_2 \dots, x_k)$, the term $B_{i_1, i_2, \dots i_j}$ refers to fault injection upon x_{i_1}, $x_{i_2} \dots x_{i_j}$. That is $x_{i_1} = x_{i_2} = \cdots = x_{i_j} = 1$. As an example:

let $x_0 = 1 \Rightarrow$
$B_0 = x_1 + x_3 + x_4 + x_3x_4 + x_1x_2 + x_2x_3 + x_2x_4 + x_1x_2x_4 + x_2x_3x_4,$
let $x_1 = 1 \Rightarrow$
$B_1 = 1 + x_4 + x_0x_3 + x_3x_4 + x_0x_2 + x_0x_2x_3 + x_0x_2x_4 + x_2x_4 + x_2x_3x_4$
$\vdots$
let $x_0 = x_1 = 1 \Rightarrow$
$B_{0,1} = x_1 + x_2 + x_4 + x_0x_3 + x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$
$\vdots$
let $x_2 = x_3 = x_4 = 1 \Rightarrow$
$B_{2,3,4} = 1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4$

C. Hao's Method

In 2007, Hao [13] introduced sufficient and necessary conditions of the existence of low degree multiplies for a given Boolean function f is analyzed and three algorithms to find annihilators, g of a Boolean function f. We consider all the n variable non-zero monomials of degree $\leq d$ denoted by:

$$A_d = 1, x_1, x_2, ..., x_n, x_1x_2, x_1x_3, ..., x_{n-1}x_n, ..., x_{n-d+1}x_{n-d+2}...x_n$$
$$= p_1p_2p_3, ..., p_r (r = \sum_{i=0}^{d} C_n^i)$$

Theorem 1. $|C| < A_d \Rightarrow$ *There exists at least one annihilator of* f *with degree* $\leq d$.

Theorem 2. *There exists annihilator of* f *with degree* $d \Leftrightarrow rank(M_d(h)) < |A_d|$.

Algorithm 1 [8]: *Given a n-variable Boolean function* f *, find all annihilators of* f *with degree* $\leq d$.

1) Step 1: Construct matrix $M_d(f)$.
2) Step 2: Convert $M_d(f)$ into row ladder matrix $M_d(f)^*$ using Gaussian elimination.
3) Step 3: If there exist zero-rows in $M_d(f)^*$ it certainly exists an annihilator g of f and obtain g by using the inverse procession of Step 2, or else, there is no annihilator of f with degree $\leq d$.

Remark 1. *Construction of the matrix need evaluate* fp_i *on all* $x \in \{0,1\}^n$ *, and it need many computations. If Boolean function* f *is represented by a* 2^n *vector, we can abbreviate these computations.*

Theorem 3. *n-variable Boolean function* $h \equiv 0$ *, coefficients of* h *are zeroes.* [8].

Theorem 4. *There exists annihilator of f with degree* $\leq d$ *, The rows of* $N_d(f)$ *are linear dependent* $\Leftrightarrow$ *rank* $N_d(f) < |A_d|$. [8]

Algorithm 2 [8]: *Given an n-variable Boolean function* f *, find all annihilators of* f *with degree* $\leq d$.

1) Step 1: Construct matrix $N_d(f)$
2) Step 2: Convert $N_d(f)$ into row ladder matrix $N_d(f)^*$ using Gaussian elimination.
3) Step 3: If there exist zero-rows in $N_d(f)^*$, it certainly exists an annihilator g of f and obtain g by using the inverse procession of Step 2, or else, there is no annihilator of f with degree $\leq d$.

Theorem 5. *Let* f *be any Boolean function in* B_n *. Then there exists annihilator of* f *with degree* $\leq d$ *if and only if there exists* $h \in B_n$ *with degree* $\leq d$ *d such that the degree of* $(1+f) \cdot h = g \leq d$. [8]

Algorithm 3 [8]: *Given an n-variable Boolean function* f *, find all annihilators of* f *with degree* $\leq d$.

Input: *n*-variable Boolean function f

Output: Boolean function h and g with degree $\leq d$ such that $g = (1+f) \cdot h$

1) Step 1: Define

$$\sum_{i=0}^{d} C_n^i \text{ X } \sum_{i=d+1}^{n} C_n^i \quad (2)$$

2) Step 2: Convert $U_d(f)$ into row ladder matrix $U_d(f)^*$ using Gaussian elimination.
3) Step 3: If there exists zero-rows in $U_d(f)^*$ it certainly exists $h \in B$

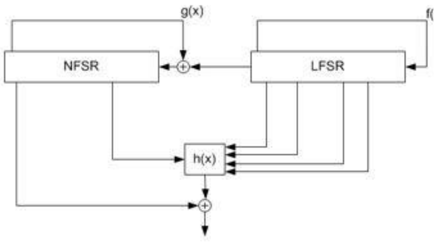
with degree $\leq d$ such that the degree of $(1+f)\cdot h = g$ less than d and we can obtain h and g using the inverse procession of Step 2, or else there is no annihilator of f with degree $\leq d$.

This algorithm 3 can generate all annihilators (with degree $\leq d$) of both f and $(1+f)$.

IV. DESCRIPTION OF GRAIN V0

Grain v0 stream cipher was developed by [14] and the design was targets hardware that only have a very limited memory, limited power consumption and limited gate count. This algorithm was established on only two shift registers and one non-linear filter function namely an LFSR, an NFSR and a filter function as shown in Fig. 1.

Fig. 1. Structure of Grain v0 Stream Cipher



A. Design of Grain v0

The content of LFSR is denoted as $s_i, s_{i+1}, s_{i+2}, \ldots, s_{i+79}$ meanwhile content of NFSR denoted as $b_i, b_{i+1}, b_{i+2}, \ldots, b_{i+79}$. The LFSR $f(x)$ feedback polynomial is a primitive 80 degree polynomial and is defined as:

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80} \quad (3)$$

and this the update function LFSR to remove any possible ambiguity:

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+13} + s_i \quad (4)$$

The feedback polynomial of the NFSR, $g(x)$, shall be described as:

$$g(x) = 1 + x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59} \quad (5)$$

and this is NFSR update feature to eliminate any ambiguities: (including bit s_i that masked with the input in below function)

$$b_{i+80} = s_i + b_{i+63} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+15} + b_{i+9} + b_{i+63}b_{i+60} + b_{i+33}b_{i+37} + b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21} \quad (6)$$

B. Grain v0 Boolean function

Grain v0 Boolean function is given by;

$$h(x) = x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4 \quad (7)$$

Let $n = 5$ and $d = 3$ in the Grain v0.

V. FAULT INJECTION ANALYSIS ON BOOLEAN FUNCTION OF GRAIN V0

As mention in previous section, Grain v0's Boolean function is in equation 7. We will inject value of one (1) as fault value into each of active coefficient. In Grain v0, we obtained nineteen (19) active coefficients.

Let new generated Injected Boolean function of Grain v0 is as below (refer subsection III-B):

Let $x_0 = 1$
$x_1 + x_4 + x_3 + x_3x_4 + x_1x_2 + x_2x_3 + x_2x_4 + x_1x_2x_4 + x_2x_3x_4$ (8)

Let $x_1 = 1$
$1 + x_4 + x_0x_2 + x_0x_3 + x_2x_4 + x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_2x_3x_4$ (9)

Let $x_2 = 1$
$x_1 + x_4 + x_0x_1 + x_0x_4 + x_1x_4$ (10)

Let $x_3 = 1$
$x_0 + x_1 + x_0x_2 + x_2x_4 + x_0x_1x_2 + x_0x_2x_4 + x_1x_2x_4$ (11)

Let $x_4 = 1$
$1 + x_1 + x_3 + x_0x_2 + x_0x_3 + x_1x_2 + x_2x_3 + x_0x_1x_2 + x_0x_2x_3$ (12)

Let $x_0x_1 = 1$
$x_1 + x2 + x_4 + x_0x_3 + x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$ (13)

Let $x_0x_2 = 1$
$x_3 + x_0x_3 + x_3x_4 + x_1x_2x_4 + x_2x_3x_4$ (14)

Let $x_0x_3 = 1$
$1 + x_1 + x_2 + x_4 + x_3x_4 + x_0x_1x_2 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$ (15)

Let $x_0x_4 = 1$
$x_1 + x_2 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_4 + x_2x_3x_4$ (16)

Let $x_1x_2 = 1$
$x_0 + x_1 + x_0x_3 + x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_2x_3x_4$ (17)

Let $x_1x_4 = 1$
$x_1 + x_2 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_2x_3x_4$ (18)

Let $x_2x_3 = 1$
$x_0 + x_1 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_4 + x_1x_2x_4$ (19)

Let $x_2x_4 = 1$
$x_0 + x_3 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3$ (20)

Let $x_3x_4 = 1$
$1 + x_1 + x_2 + x_4 + x_0x_3 + x_0x_1x2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4$ (21)

Let $x_0x_1x_2 = 1$
$1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$ (22)

Let $x_0x_2x_3 = 1$
$1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$ (23)

Let $x_0x_2x_4 = 1$
$1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_1x_2x_4 + x_2x_3x_4$ (24)

Let $x_1x_2x_4 = 1$
$1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_2x_3x_4$ (25)

Let $x_2x_3x_4 = 1$

$$1 + x_1 + x_4 + x_0x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 \quad (26)$$

From the analysis via FIA and HAO's algorithm on Grain v0, we achieved possibility of annihilator(s). Consequently, we can obtain that only six (6) active coefficients in Boolean injected function generated zero row in *Md**. The coefficients involved that generated with one (1) zero row in *Md** is:

1) x_0,
2) x_1,
3) x_2,
4) x_4,

Meanwhile we get two (2) zero rows in *Md** each for injected into coefficients, x_3. We also obtained four (4) zero rows in *Md** by injected into coefficients x_0x_2.

VI. ILLUSTRATION OF REDUCING BOOLEAN FUNCTION DEGREE VIA NEWLY FOUND ANNIHILATORS

For this section, we will illustrate our result on Grain v0 by using **Theorem 5**. We achieved six matrices, Md^* that have zero row(s) when we injected the fault value via coefficients x_0, x_1, x_2, x_3, x_4 and x_0x_2, but only injection Boolean function via x_4 produces annihilators.

For the case x_1, we achieved $f = 1+x_4+x_0x_2+x_0x_3+ x_2x_4+x_3x_4+x_0x_2x_3+ x_0x_2x_4 + x_2x_3x_4$. The corresponding annihilator $g = x_2 + x_1x_2 + x_1x_3 + x_2x_4$ did not reduce the complexity to find the initial key string of the injected Boolean f; of the form $1 + f$. We observed $(1+f)\ g = h = x_0x_2(1 + x_1 + x_4 + x_1x_4) + x_0x_1x_3(1 + x_2x_4)$. The degree of h is 2 and is the same as $(1 + f)$.

For the case x_2, we achieved $f = x_1 + x_4 + x_0x_1 + x_0x_4 + x_1x_4$. The corresponding annihilator $g = x_0x_4 + x_2x_4$ did not reduce the complexity to find the initial key string of the injected Boolean f; of the form $1 + f$. We observed $(1+f)\ g = h = x_0x_4(1 + x_1 + x_2 + x_1x_2)$. The degree of h is 2 and is the same as $(1 + f)$.

Next, when we injected the Grain v0 Boolean function via x_4, we obtained $f = 1+x_1+x_3+x_0x_2+x_0x_3+x_1x_2 + x_2x_3 + x_0x_1x_2 + x_0x_2x_3$. The corresponding annihilator is $g = x_0x_1 + x_1x_2$. Observe that $(1 + f) \bullet g = h = (x_0x_1) + (x_0x_1x_2) = x_0x_1(1 + x_2) = u1\ \ u2$ where $u_1 = (x_0x_1)$ and $u_2 = (1 + x_2)$.

For the case $(1 + f) = 1$, we assumed $u_1 = 1$ and $u_2 = 1$, and obtained **Table I**. It shows that in our case the complexity of guessing the initial key bit is $2^0 = 1$. This is a reduction from the complexity of $2^4 = 16$ upon the published Grain v0 Boolean function.

We obtained one first degree and one second degree simultaneous equation instead of third degree equation. If $(1 + f) = 1$, then we have few combinations of $u_1 u_2 = 1$. We assume that $u_1 = 1$ and $u_2 = 1$ and we generate **Table I** and managed to get only $2^0 = 1$ complexity of guessing compared with the published Boolean of Grain v0 that has $2^4 = 16$ complexity to guessing initial key bit.

TABLE I: Grain v0 - Combination for (1 + f) = 1

x_0	x_1	x_2	$(1 + f)$
1	1	0	1

For the case $(1 + f) = 0$, we assumed either $u_1 = 1$ and $u_2 = 0$ or $u_1 = 0$ and $u_2 = 1$ or $u_1 = 0$ and $u_2 = 0$ and obtained **Table II**. It shows that in our case the complexity of guessing initial key bit is 7. This is reduction from the complexity of $2^4 = 16$ upon the published Grain v0 Boolean function.

TABLE II: Grain v0 - Combination for $(1 + f) = 0$

x_1	x_2	x_3	$(1 + f)$
1	1	0	0
1	0	1	0
0	1	1	0
1	0	0	0
0	1	0	0
1	0	1	0
0	1	1	0

VII. DISCUSSION

From the analysis and results we have generated eighteen injected Boolean functions and successfully obtained three possible annihilator(s) from Grain v0's Boolean function via FIA with Hao's method. We then identified that the annihilator, $g = x_0x_1 + x_1x_2$ which was obtained by injecting fault value upon x_4, had capacity to reduce the complexity of determining the initial key upon our injected Grain v0 Boolean function as showed in **Table III**. That is from complexity of (24 = 16) + (24 = 16) = 32 to (20 = 1) + 7 = 8. In conclusion this identified annihilator provided much needed information on the security of Grain v0 and will be utilized to launch algebraic attacks upon Grain v0 stream cipher.

TABLE III: Annihilator upon Grain v0's Injected Boolean Function

Coefficient	Annihilator
$x4$	$x0x1 + x1x2$

VIII. CONCLUSION

As for the conclusion, this paper successfully conducted a Fault Injection Analysis (FIA) on Boolean function of selected stream cipher such as Grain v0. For Grain v0 stream cipher, we got four coefficients that produced one zero row, one coefficient that produced two zero row and one coefficient that produced four zero row. But only three of this output generate possible annihilators as in Section VII; $x2 + x1x2 + x1x3 + x2x4$, $x0x4 + x2x4$ and $x0x1 + x1x2$. So, from eighteen generated injected Boolean function, we only found three annihilators but only this annihilator $x0x1 + x1x2$ manage to reduce degree and complexity of published Boolean function.

IX. FUTURE WORKS

We planned to do analysis for another algorithms that have more complicated Boolean function as Grain v1 or Grain-128 and Rakaposhi algorithms. Hopefully, we can manage to get funding to conduct future research.

X. REFERENCES

[1] H. Delfs, and H. Knebl, :Introduction to cryptography, Berlin etc.: Springer, vol. 2, pp.11-48, 2002.
[2] R.A. Rueppel, "*Analysis and design of stream ciphers*", *Springer Science & Business Media*, 2012
[3] K.M. Martin, "Everyday cryptography" *The Australian Mathematical Society*, 231(6), 2012.

[4] G. Banegas, Attacks in Stream Ciphers: A Survey. *IACR Cryptology ePrint Archive*, pp.677, 2014.

[5] C. Carlet, Boolean functions for cryptography and error correcting codes, *Boolean models and methods in mathematics, computer science, and engineering*, 2, pp.257-397, 2010.

[6] N.T. Courtois, "Fast algebraic attacks on stream ciphers with linear feedback", *In Annual International Cryptology Conference Springer*, Berlin, Heidelberg, pp. 176-194, August 2003.

[7] W. Meier, E. Pasalic, and C. Carlet, "Algebraic attacks and decomposition of Boolean functions". *In International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, Berlin, Heidelberg. pp. 474-491, May 2004.

[8] Zhang, Haina, and Xiaoyun Wang. "Cryptanalysis of Stream Cipher Grain Family." *IACR Cryptology ePrint* Archive (2009): 109, 2009.

[9] Subhadeep Banik, Subhamoy Maitra, and Santanu Sarkar. "A differential fault attack on the grain family of stream ciphers." *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, Berlin, Heidelberg, 2012.

[10] D. Roy, P. Datta and S. Mukhopadhyay, Algebraic cryptanalysis of stream ciphers using decomposition of Boolean function. *Journal of Applied Mathematics and Computing*, 49(1-2), pp.397-417, 2015.

[11] A. Barenghi, L. Breveglieri, I. Koren and D. Naccache, 2012. "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures". *Proceedings of the IEEE*, 100(11), pp.3056-3076, 2012.

[12] F. Armknecht, Algebraic Attacks and Annihilators. *In WEWoRC* (pp. 13-21), 2005.

[13] C. Hao, W. Shimin, and Z. Zepeng, "Several algorithms to find annihilators of Boolean function". In The First International Symposium on Data, Privacy, and E-Commerce (ISDPE 2007) IEEE pp. 341- 343, November 2007.

[14] Martin Hell, Thomas Johansson, and Willi Meier. "Grain: a stream cipher for constrained environments". *International Journal of Wireless and Mobile Computing* 2.1: 86-93, 2007.

[15] F. Kong, G. Yang, H. Liu, Y. Jiang, C. Hu, and D. Zhou, "Fault-injection Attack and Improvement of a CRT-RSA Exponentiation Algorithm". In *Proceedings of the 2019 the 9th International Conference on Communication and Network Security* (pp. 123-127, November 2017.