Strengthening Crypto-1 Cipher Against Algebraic Attacks

Farah Afianti & Ari M. Barmawi

School of Computing, Telkom University,
Jl. Telekomunikasi 1, Terusan Buah Batu Bandung, 40257 Indonesia
Email: farahaphie@tass.telkomuniversity.ac.id

Abstract. In the last few years, several studies addressed the problem of data security in Mifare Classic. One of its weaknesses is the low random number quality. This causes SAT solver attacks to have lower complexity. In order to strengthen Crypto-1 against SAT solver attacks, a modification of the feedback function with better cryptographic properties is proposed. It applies a primitive polynomial companion matrix. SAT solvers cannot directly attack the feedback shift register that uses the modified Boolean feedback function, the register has to be split into smaller groups. Experimental testing showed that the amount of memory and CPU time needed were highest when attacking the modified Crypto-1 using the modified feedback function and the original filter function. In addition, another modified Crypto-1, using the modified feedback function and a modified filter function, had the lowest percentage of revealed variables. It can be concluded that the security strength and performance of the modified Crypto-1 using the modified feedback function and the modified filter function are better than those of the original Crypto-1.

Keywords: *Boolean function; Crypto-1; primitive polynomial; pseudorandom generator; SAT solver.*

1 Introduction

Recently the use of RFID systems has increased significantly, so that several companies produce RFID technology, among which NXP Semiconductors. One of the best-known products of NXP Semiconductors is Mifare Classic. In 2009, this product covered more than 70% of the contactless smartcard market [1]. However, there is a problem with Mifare Classic's data security.

In the last few years, several studies have addressed the problem of data security in Mifare Classic [1]-[5]. One of the attacking processes discussed by Curtois, *et al.* [2] uses algebraic attacks to recover the secret key in a short time without accessing the hardware. The research reported by Liu, *et al.* [6] showed how to retrieve a Crypto-1 secret key even faster than Curtois, *et al.* [2].

The objective of the algebraic attack in [6] was to test the quality of the pseudorandom numbers [2]. Furthermore, the attacker may recover the secret

Received January 30th, 2015, 1st Revision March 29th, 2015, 2nd Revision May 20th, 2015, Accepted for publication June 4th, 2015.

key by solving a large number of polynomial equations. There are several ways to solve such equations, such as linearization, Gröbner bases or extended linearization (XL) algorithms [7], SAT solvers [8], etc. SAT solvers are efficient [8] because they are faster than exhaustive search and do not need additional memory to store the CNF SAT problem [9]. They solve the equation by first converting the polynomial into a CNF SAT problem and then guessing each variable using the SAT solver algorithm.

Since Mifare Classic is vulnerable to attacks, in 2008, NXP semiconductors introduced a new product called Mifare Plus to replace Mifare Classic. However, Karsten Nohl [10] found that Crypto-1 is vulnerable to algebraic attacks both in Mifare Classic and Mifare Plus. NXP Semiconductors developed another cipher for the RFID transponder system for the car immobilizer industry called Hitag2. It is claimed to be more secure than Crypto-1 because the taps of the Boolean function of Hitag2 are not regular [11]. Even though Hitag2 has higher algebraic immunity than Crypto-1 it can still be cracked using an SAT solver in 6.5 hours [12].

This paper proposes a modification of the Boolean feedback function in the pseudorandom generator of Crypto-1 for strengthening Crypto-1 against SAT solver attacks. For this purpose, the cryptosystem should satisfy cryptographic properties [6] and the Boolean function should satisfy cryptographic properties as well. The proposed modification of the Boolean feedback function uses a primitive polynomial companion matrix derived from Wang’s construction [13]. Wang’s construction uses the same construction as the Carlet-Feng function [14]. Both can be classified as perfect algebraic immune functions [6] that meet the cryptographic properties requirement. However, the Carlet-Feng function cannot be implemented efficiently with large input variables because of the discrete logarithm problem [13]. This is the reason why we prefer to adopt Wang’s construction rather than the Carlet-Feng function, although both of them are perfect algebraic immune functions.

This rest of this paper is organized as follows. Section 2 discusses Crypto-1 and SAT solver attacks. Section 3 describes the cryptographic properties and Wang’s construction. Section 4 briefly explains the experimental testing and discusses the algorithmic complexities of the proposed method for overcoming SAT solver attacks. Finally, the conclusion is given in Section 5.

2 Attacking Crypto-1 using SAT solvers

Crypto-1 can be attacked using SAT solvers [2],[12]. SAT solvers are able to recover the secret key of Crypto-1 without interacting with the hardware (both

RFID tag and reader). Section 2.1 and 2.2 discuss Crypto-1 and the method for attacking Crypto-1.

2.1 Crypto-1

Crypto-1 is a proprietary stream cipher produced by NXP Semiconductors. The input of the Crypto-1 cryptosystem is a message or plaintext and the output is a ciphertext with the same length as the plaintext. The ciphertext is obtained by XOR-ing the keystream and the plaintext, where the keystream is generated using a linear feedback shift register (LFSR).

The basic concept of the Crypto-1 encryption method is as follows. The Crypto-1 cipher encrypts the authentication message and other messages exchanged between the RFID tag and reader. Crypto-1 uses a 48-bit LFSR and the generator polynomial of Crypto-1 is as shown in Eq. (1) [3].

$$g(x) = x^{48} + x^{43} + x^{39} + x^{38} + x^{36} + x^{34} + x^{33} + x^{31} + x^{29} + x^{24} + x^{23} + x^{21} + x^{19} + x^{13} + x^{9} + x^{7} + x^{6} + x^{5} + 1 \qquad (1)$$

The initialization input of this LFSR is the secret key of the Mifare RFID tag. The binary LFSR of size n is a finite-state automaton with an internal state of n bits. In each clock cycle, the feedback function L shifts the state by one position, where the input bit is a linear function of the previous bits [15]. This means that the LFSR shifts one bit to the left and replaces the rightmost tap using a feedback function. After this, one bit of the keystream is calculated by using a filter function. These processes are repeated and terminated when the length of the keystream is the same as the length of the plaintext. The details are shown in Figure 1.

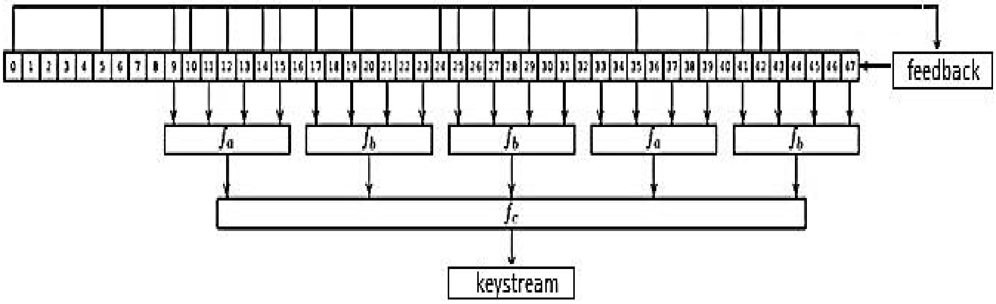


Figure 1 48-bit LFSR of Crypto-1 cryptosystem.

At time i, the internal state of the LFSR is $a_i \ldots a_{i+47}$ and at time(i+1) the internal state is $a_{i+1} \ldots a_{i+48}$. Eq. (2) is the feedback function to calculate the new rightmost tap register [1].

$$a_{i+48} = a_i \oplus a_{i+5} \oplus a_{i+9} \oplus a_{i+10} \oplus a_{i+12} \oplus a_{i+14} \oplus a_{i+15} \oplus a_{i+17}$$
$$\oplus a_{i+19} \oplus a_{i+24} \oplus a_{i+27} \oplus a_{i+29} \oplus a_{i+35} \oplus a_{i+39} \oplus a_{i+41}$$
$$\oplus a_{i+42} \oplus a_{i+43} \oplus another\_input \quad (2)$$

The keystream as the output of the LFSR is calculated using a filter function. The filter function $f : F_2^{48} \to F_2$ is defined by Eq. (3) [3].

$$f(x_0 x_1 \ldots x_{47}) = f_c(f_a(x_9, x_{11}, x_{13}, x_{15}), f_b(x_{17}, x_{19}, x_{21}, x_{23}), f_b (x_{25}, x_{27}, x_{29}, x_{31}), f_a(x_{33}, x_{35}, x_{37}, x_{39}), f_b(x_{41}, x_{43}, x_{45}, x_{47})) \quad (3)$$

where $f_a, f_b : F_2^4 \to F_2$ and $f_c : F_2^5 \to F_2$ are defined by Eq. 4 [1].

$$f_a(y_0, y_1, y_2, y_3) = ((y_0 \vee y_1) \oplus (y_0 \wedge y_3)) \oplus (y_2 \wedge ((y_0 \oplus y_1) \vee y_3)) \quad (4.a)$$

$$f_b(y_0, y_1, y_2, y_3) = ((y_0 \wedge y_1) \vee y_2) \oplus ((y_0 \oplus y_1) \wedge (y_2 \vee y_3)) \quad (4.b)$$

$$f_c(y_0, y_1, y_2, y_3, y_4) = (y_0 \vee ((y_1 \vee y_4) \wedge (y_3 \oplus y_4))) \oplus ((y_0 \oplus (y_1 \wedge y_3)) \wedge ((y_2 \oplus y_3) \vee (y_1 \wedge y_4))) \quad (4.c)$$

The algebraic normal forms (ANF) of f_a , f_b and f_c are in Eq. (5).

$$f_a(y_0,y_1,y_2,y_3) = y_0y_2y_3+y_1y_2y_3+y_0y_1+y_0y_2+y_0y_3+y_1y_2+y_2y_3 +y_0+y_1 \quad (5.a)$$

$$f_b(y_0,y_1,y_2,y_3) = y_0y_1y_2+ y_0y_2y_3 +y_1y_2y_3 +y_0y_1 +y_0y_2 +y_0y_3 +y_1y_2 +y_1y_3+y_2 \quad (5.b)$$

$$f_c(y_0,y_1,y_2,y_3,y_4)= y_0y_1y_2y_4 +y_1y_2y_3y_4 +y_0y_1y_3 +y_0y_1y_4+ y_1y_2y_3 + y_0y_3y_4+y_1y_3y_4+y_0y_2+y_0y_3+y_0y_4+y_3y_4+y_0+y_4 \quad (5.c)$$

Based on the encryption method of Crypto-1 the author claims that it is secure.

2.2 SAT Solver Attacks

SAT stands for ‘satisfiability’. A Boolean formula is satisfiable if there are several value assignments of false and true to its variables that cause it to be evaluated to true [16]. In contrast, if it is evaluated to false then it is called unsatisfiable. The input of an SAT problem must be in conjunctive normal form (CNF) so that it is called a CNF-SAT problem. CNF-SAT consists of several clauses. The clauses can be seen as a logical expression connected with a conjunctive (AND-gate) expression. Every clause is a large disjunction (OR-gate) of several variables called literals [9]. An SAT solver is an algebraic method to solve all of the system’s polynomial equations over finite fields [17].

Algebraic attacks are passive. They work independently of the quality of the random numbers [2]. The principle of the algebraic attack is converting the cipher into polynomial equations and solving those equations. The solution is used to recover the secret key of the cipher [9]. Based on [15], the LFSR system can be written as Eq. (6).

$$f(L^i(a))=b^i \tag{6}$$

where : $f()$ is a filter function
$L()$ is a feedback function
a is an internal state of the LFSR
b^i is a keystream from the internal state of the LFSR at time i

The LFSR in the Crypto-1 cipher can be converted into polynomial equations so that it can be attacked algebraically by solving these equations. The basic concept of attacking Crypto-1 is using several outputs of the keystream as the input of the SAT solver. The keystream can be expressed by equations in algebraic normal form (ANF). Before the equation can be solved using an SAT solver, the ANF must be converted into CNF. This study focuses on an over defined system where the number of equations is greater than the number of unknown variables. The details of how an algebraic attack works are shown in Figure 2.

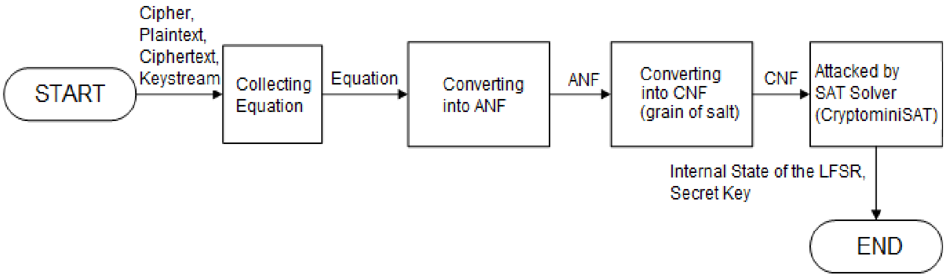

Cipher,
Plaintext,
Ciphertext,
Keystream
START
Collecting Equation
Equation
Converting into ANF
ANF
Converting into CNF (grain of salt)
CNF
Attacked by SAT Solver (CryptominiSAT)
Internal State of the LFSR, Secret Key
END

Figure 2 Implementation of algebraic attack.

Algebraic cryptanalysis is implemented by building the equations from the cipher that will be attacked. It is supported by the ciphertext, the plaintext and the cipher. The keystream can be obtained by XOR-ing the ciphertext with the plaintext. Since the cipher is already known, the register taps can be recovered from the last clock until the first clock. The main purpose of an algebraic attack is obtaining the secret key that can be represented as the content of the initial register tap in the first clock. All of the constraints in each clock cycle are used to build the equation. After obtaining the equation, they should be converted into ANF, followed by converting the ANF into CNF. This can be done automatically using the Grain of Salt tool [18]. Finally, an attack using the CryptoMiniSat tool [12] is conducted. The output of the tool is the internal state of the feedback shift register and other parameters to analyze the complexity during the attacking process. The complexity of CryptoMiniSat can be concluded from the amount of memory and CPU time used [17].

Analyzing the results of SAT solver attacks against Crypto-1, it can be concluded that the vulnerability of Crypto-1 is caused by the weak feedback function against algebraic attacks.

3 Modified Crypto-1

Since the feedback function is vulnerable to algebraic attacks, the main objective of this research is strengthening Crypto-1 by replacing the original feedback function with one that has better cryptographic properties (see Section 3.1). For this purpose, Wang's construction is used because it has the best values – near optimal– for all cryptographic properties. Wang's construction is explained in detail in Section 3.2.

The construction process of the modified Boolean feedback function is shown in Figure 3.

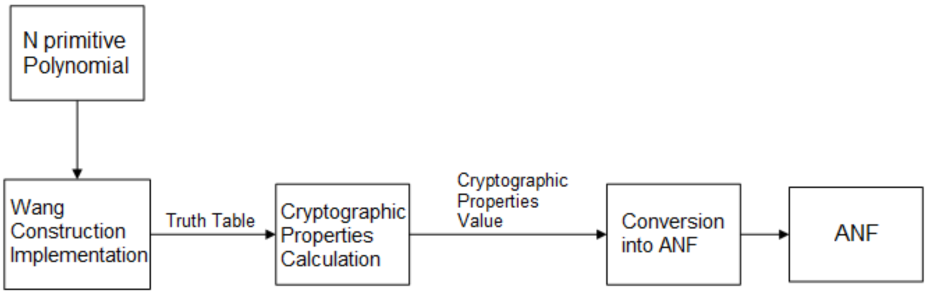

N primitive Polynomial
Wang Construction Implementation
Truth Table
Cryptographic Properties Calculation
Cryptographic Properties Value
Conversion into ANF
ANF

Figure 3 Method for finding the new Boolean function.

The input of Wang's construction is a primitive polynomial and the output is a truth table. The cryptographic properties, such as algebraic degree, algebraic immunity, nonlinearity and balancedness, can be calculated from the truth table. These parameters should be calculated for further use, such as cryptographic properties evaluation or comparison between the original and the modified function for deciding whether the modified function works better against SAT solver attacks. Calculation of the cryptographic properties is discussed in detail in Section 3.1. The input of this process is the truth table from Wang's construction and the output are the cryptographic property values.

Since the truth table containing the cryptographic property values cannot be used directly, it has to be converted into a Boolean function. ANF is the standard form of Boolean functions. It can be computed from the truth table using the fast Möbius transform.

3.1 Cryptographic Properties

The cryptographic properties are used as the basic criteria to find the best Boolean function. This is necessary for defending the system against algebraic attacks. There are several cryptographic properties in nonlinear Boolean functions, such as:

a. Algebraic Degree

A good Boolean function has a high algebraic degree. The algebraic degree is the maximum number of ANF clauses in a Boolean function that are written as Eq. (7) or can be rewritten as Eq. (8) [6].

$$deg(f) = max\ wt(c)|a_c \neq 0 \tag{7}$$

$$f(x) = \sum_{c \in F_2^n} a_c x^c, a_c \in F_2 \tag{8}$$

The Hamming weight of *f* denoted by *wt*(*f*) is the number of ones (nonzero) in the truth table of *f*. The way to calculate *wt*(*f*) is as follows:

Let F_2 denote Galois Field (2) and F_2^n denote the *n*-dimensional binary vector space over F_2. The mapping of the Boolean function $f : F_2^n \rightarrow F_2$ is denoted as FB_n. The support of *f* is calculated by Eq. (9) [6].

$$support(f) = x \in F_2^n\ |f(x) = 1 \tag{9}$$

Furthermore, the Hamming distance between *n*-variable Boolean function *f* and *g* is the number of $x \in Fn$ where $f(x) = g(x)$ and is denoted by $d(f,g) = wt(f + g)$. The affine function is a Boolean function that has a degree of at most one. The general form of affine functions is shown in Eq. (10) [6].

$$f(x) = c_0 \oplus c_1 x_1 \oplus \ldots \oplus c_n x_n \tag{10}$$

A cryptosystem using a Boolean function for randomization can be attacked easily if its degree is low [19].

b. Balancedness

The Boolean function *f* is balanced if the truth table contains an equal number of zeros and ones or can be calculated using Eq. (11) [6].

$$wt(f) = 2^{n-1} \tag{11}$$

Balancedness means that the output of the function is uniformly distributed.

c. Nonlinearity

The nonlinearity of the Boolean function *f* denoted by *Nl*(*f*) is the minimum Hamming distance between *f* and all affine functions [19]. The higher the value of the nonlinearity, the higher the randomness is. The nonlinearity can be calculated using Eq. (12) [19].

$$Nl(f) = 2^{n-1} - \frac{1}{2} \max_{\alpha \in F_2^n} \{|W_f(\alpha)|\} \tag{12}$$

where W_f is the Fourier or Walsh-Hadamard transform of function *f* .

The Fourier or Walsh-Hadamard transform can be calculated using Eq. (13) [19].

$$W_f(\alpha) = \sum_{x \in F_2^n} (-1)^{f(x) \oplus \alpha.x} \tag{13}$$

In simple terms, the Walsh transform is the matrix vector product of matrix $W_f\,t$ and Boolean function *f*(*x*)[20]. Matrix $W_f\,t$ is an *n* x *n* matrix from which the value of $W_f\,t$ can be calculated using Eq. (14) [19].

$$W_f t(0) = 1$$

$$W_f t(N) = \begin{pmatrix} W_f t(N-1) & W_f t(N-1) \\ W_f t(N-1) & -W_f t(N-1) \end{pmatrix} \tag{14}$$

where $N = 2^n$

The upper bound of the nonlinearity can be calculated with Eq. (15) [19].

$$Nl(bent = n) = 2^{n-1} - 2^{n/2 - 1}, n > 2 \tag{15}$$

where *n* is the number of input variables.

The function that reaches the upper bound in Eq. (15) is called the bent function.

d. Algebraic Immunity

There are several types of algebraic attacks. Some of them are algebraic attacks that can be defended against by algebraic immunity; fast algebraic attacks that can be defended against by algebraic immunity; and probability algebraic attacks. Cryptosystems using a Boolean function with perfect algebraic immunity are resistant against those types of algebraic attacks [6].

The algebraic immunity of a Boolean function f is the minimum value of e such that the product of the Boolean function g with degree at most e and f or $f + 1$ is equal to zero. This can be written as Eq. (16) [6].

$$AI(f) = min\{deg(g) | g.f = 0 \text{ or } g.(f + 1) = 0, g = 0 \text{ and } g \in FB_n\} \qquad (16)$$

The optimal algebraic immunity of a Boolean function with n variables is $\left\lceil \frac{n}{2} \right\rceil$[6].

A perfect algebraic immune function must have maximum AI and agrees with the following definition [6]:

> *Let f be an n-variable Boolean function. The function f is said to be perfect algebraic immune (PAI) if for any positive integer $e < n/2$, the product gf has degree at least $n - e$ for any non-zero function g of degree at most e.*

The number of input variables in a balanced Boolean function with perfect algebraic immunity is equal to a power of two plus one ($2^s + 1$) and the number of input variables in an unbalanced Boolean function with perfect algebraic immunity is a power of two (2^s) [6].

3.2 Wang's construction

As explained in Section 1, Wang's construction was chosen because it can be implemented in embedded hardware that has limitations in power and computation. The basic operation in embedded hardware is just 'XOR' and 'AND' and the output of Wang's construction is a truth table, which can be converted into ANF. ANF is the basic form of Boolean functions. It has only 'XOR' and 'AND' operators.

In 2010, Wang, *et al.* [13] studied the construction of Boolean functions using a primitive polynomial as the input. The construction has good cryptographic properties. Its basic concept can be explained as follows:

Let $p(x) = x^n + c_{n-1}x^{n-1} + ... + c_1x + 1$ be the primitive polynomial over $F_2{}^n$ .The companion matrix is defined as matrix A in Eq. (17) [13].

$$A = \begin{pmatrix} 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & c_1 \\ 0 & 1 & \ldots & 0 & c_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 & c_{n-1} \end{pmatrix} \tag{17}$$

Matrix A with n columns and n rows ($n \times n$ matrix) has characteristics as follows:

a. The (n –1) x (n−1) matrix, which (n–1) columns start from the first column and (n –1) rows start from the second row, is the identity matrix.
b. The last column is the coefficient of the primitive polynomial.
c. The remainder of the first rows is filled by zeros.

Given any initial value b_1, the iterative sequences of b can be defined as $B = \{b_i \mid 1 \leq i \leq 2^n - 1\}$ where the value of b_i is explained in Eq. (18) [13].

$$\begin{gathered} b_i \in F_2^n \\ b_{i+1} = Ab_i, 1 \leq \mathrm{i} \leq 2^n - 1 \end{gathered} \tag{18}$$

Construction 1 [13]. $f(x)$ *is a Boolean function on* F_2^n *and* $1_f = \{A^i\, b_1 | 0 \leq i < 2^{n-1}\}$*, where* $0 \neq b_1 \in F_n$*. Then it must be a balanced Boolean function since* $wt(f) = 2^{n-1}$.

This construction has a structure similar to the Carlet-Feng function [14], including perfect algebraic immunity if the input variables are a power of two [6].

3.3 Implementation of Wang's construction

Wang's construction is proposed as the main concept to create a Boolean function that has optimal cryptographic properties. The first step is choosing the number of input variables. Furthermore, the number of primitive polynomials is calculated from those input variables. The number of primitive polynomials over F_2^n is calculated using Eq. (19).

$$\frac{\varphi(2^n - 1)}{n} \tag{19}$$

where: n is the number of input variables
φ is Euler's totient function

Among the primitive polynomial alternatives, the one must be selected that has the highest nonlinearity. Nonlinearity is the only determined parameter property because it is stated in Wang, *et al.* [13] that the other properties (excluding nonlinearity) have the same value on all of the primitive polynomials, which is near the optimal value. How Wang's construction works is shown in Figure 4.

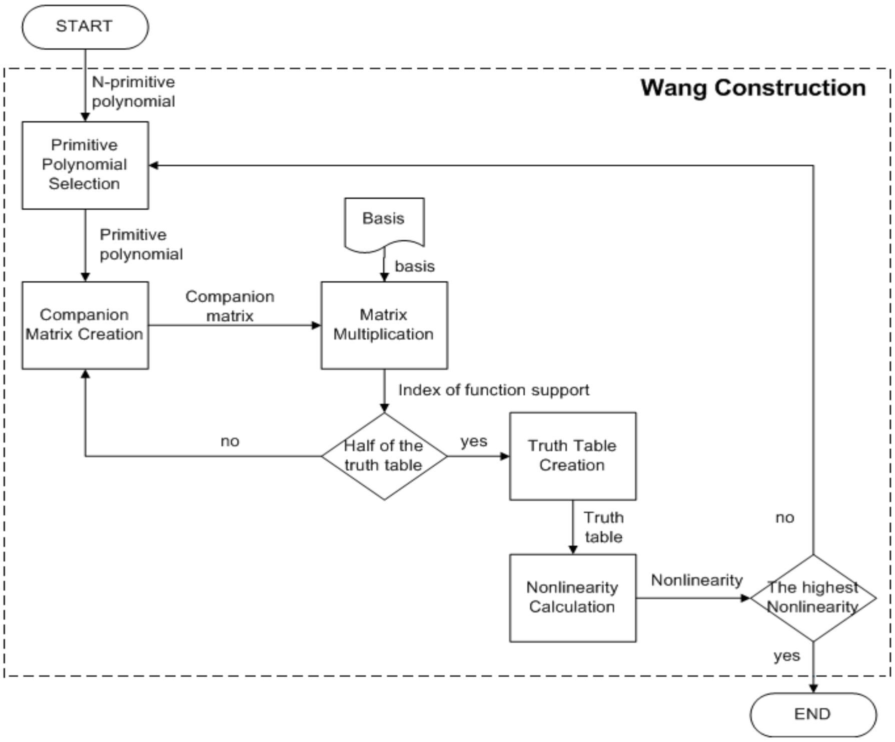


Figure 4 Procedure for creating Wang's construction.

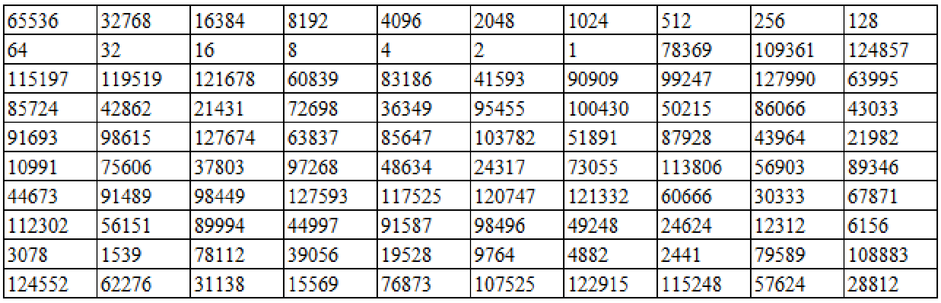
3.4 Implementation of Modified Crypto-1 Feedback Function

Wang's construction is a balanced function so that the input must be a power of two. It was chosen because it fulfills the perfect algebraic immunity requirement as mentioned in Section 3.1. Crypto-1 has 48 bits so that the choice of the input variables ($2^n + 1$) is 17 or 33. The number of input messages used in the implementation was seventeen. This number was selected because for 33 input messages a minimum RAM of (2^n x n) bytes is needed to create the truth table, which is not possible using a common workstation. The seventeen input variables over F_2^{17} had 7710 choices of primitive polynomials. This is the outcome of Eq. (19), where the value of n is 17. Among all of the primitive polynomials, two that had the highest nonlinearity, near the optimal value (bent function). These are stated in Eq. (20) and Eq. (21).

$$g(x)=x^{17}+x^{14}+x^{13}+x^{10}+x^{6}+x+1 \tag{20}$$

$$g(x)\ =x^{17}+x^{16}+x^{11}+x^{7}\ +x^{4}+x^{3}+1 \tag{21}$$

The nonlinearity value of these primitive polynomials is 65250. The nonlinearity of the bent function is 65354, so the difference is 104, or 0.159%. Of the two primitive polynomials with the highest nonlinearity either could be chosen because all the cryptographic parameters are the same and the number of variables is the same (seven) in both functions. This research chose Eq. (21) so that the total index of support ($f(x)$) was a half of (2^{17}) or 65536. The first hundred *support*($f\ (x)$) indexes can be seen in Figure 5.



65536	32768	16384	8192	4096	2048	1024	512	256	128
64	32	16	8	4	2	1	78369	109361	124857
115197	119519	121678	60839	83186	41593	90909	99247	127990	63995
85724	42862	21431	72698	36349	95455	100430	50215	86066	43033
91693	98615	127674	63837	85647	103782	51891	87928	43964	21982
10991	75606	37803	97268	48634	24317	73055	113806	56903	89346
44673	91489	98449	127593	117525	120747	121332	60666	30333	67871
112302	56151	89994	44997	91587	98496	49248	24624	12312	6156
3078	1539	78112	39056	19528	9764	4882	2441	79589	108883
124552	62276	31138	15569	76873	107525	122915	115248	57624	28812

Figure 5 Example of *support*($f(x)$) indexes.

Moreover, the value of these *support*($f(x)$) indexes in the truth table was evaluated to one, while the other indexes, which are not mentioned in *support*($f(x)$), were evaluated to zero. Then, the completed truth table could be converted into ANF. The output of Wang's construction with seventeen input variables was an algebraic normal form (ANF) with 65133 terms. These could be implemented in 65132 'OR' gates and 488668 'AND' gates.

4 Experimental Testing and Discussion

This section discusses experimental testing of the modified Crypto-1, sampling, data presentation from the experiments and an analysis of the output. The last part of this section discusses a summary of the results in answer to the objectives of this research.

4.1 Experimental Scenarios

Based on the objectives of this research, a number of experiments were conducted to evaluate the strength of Crypto-1 before and after modification. In order to evaluate the strength of Crypto-1, an evaluation of different combinations of the modified and original Boolean feedback function and filter function was done. Also, an experiment was conducted using several cases with different input messages to encrypt. This experiment was set up to evaluate whether the length of the message to encrypt has an effect in each feedback shift register scenario. The scenarios used in the experiment are summarized in Table 1.

Table 1 Experimental scenarios based on combinations of feedback and filter functions.

Scenario Code	**Feedback Shift Register**	
	Feedback Function	**Filter Function**
OFB-OFL	Original	Original
MFB-OFL	Modified	Original
OFB-MFL	Original	Modified
MFB-MFL	Modified	Modified

The OFB-OFL and OFB-MFL scenarios were conducted to evaluate whether the different tap positions of the filter function affected the original Boolean feedback function.

The MFB-OFL and MFB-MFL scenarios were conducted to evaluate whether the modified Boolean feedback function affects the whole cryptosystem, even when the tap positions of the filter function are changed.

The OFB-OFL scenario is the original Crypto-1. It consists of the original feedback function combined with the original filter function, as illustrated in Figure 1 under Section 2.1.

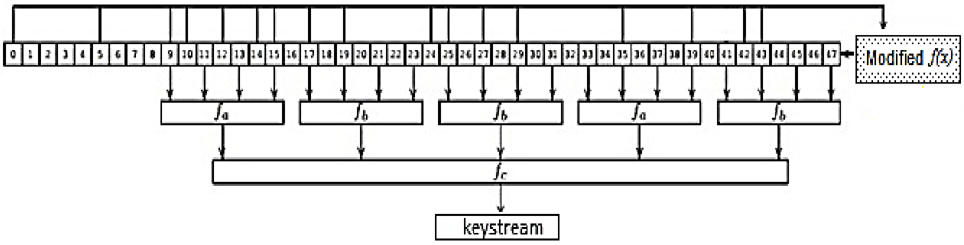


Figure 6 MFB-OFL scenario.

The MFB-OFL scenario is a modified Crypto-1 consisting of the modified feedback function combined with the original filter function, as illustrated in Figure 6.

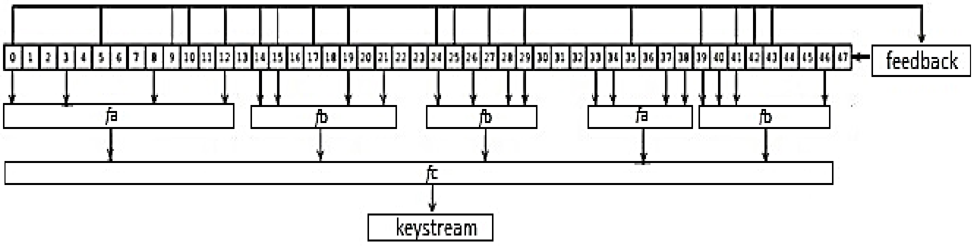


Figure 7 OFB-MFL scenario.

The OFB-MFL scenario is a modified Crypto-1consisting of the original feedback function combined with the modified filter function. The modified filter function changes the input tap positions of the filter function based on another primitive polynomial, as illustrated in Figure 7.

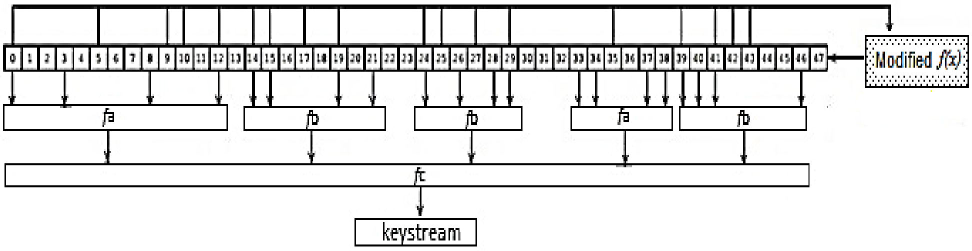


Figure 8 OFB-MFL scenario.

The MFB-MFL scenario is a modified Crypto-1 consisting of the modified feedback function combined with the modified filter function. The modified filter function in the MFB-MFL scenario is the same as that in the OFB-MFL scenario, as illustrated in Figure 8.

Updating the position of the filter function was tested to find out whether it has a substantial effect on the system. Also, the regular tap positions in the original filter function are considered a weakness [11]. One of the primitive polynomials on F_2^{48}, which has twenty one variables, can be seen in Eq. (22).

$$g(x)=x^{48}+x^{45}+x^{40}+x^{36}+x^{34}+x^{33}+x^{29}+x^{27}+x^{24}+x^{22}+x^{20}+ \\ x^{19}+x^{15}+x^{14}+x^{11}+x^{10}+x^{9}+x^{8}+x^{7}+x^{2}+1 \tag{22}$$

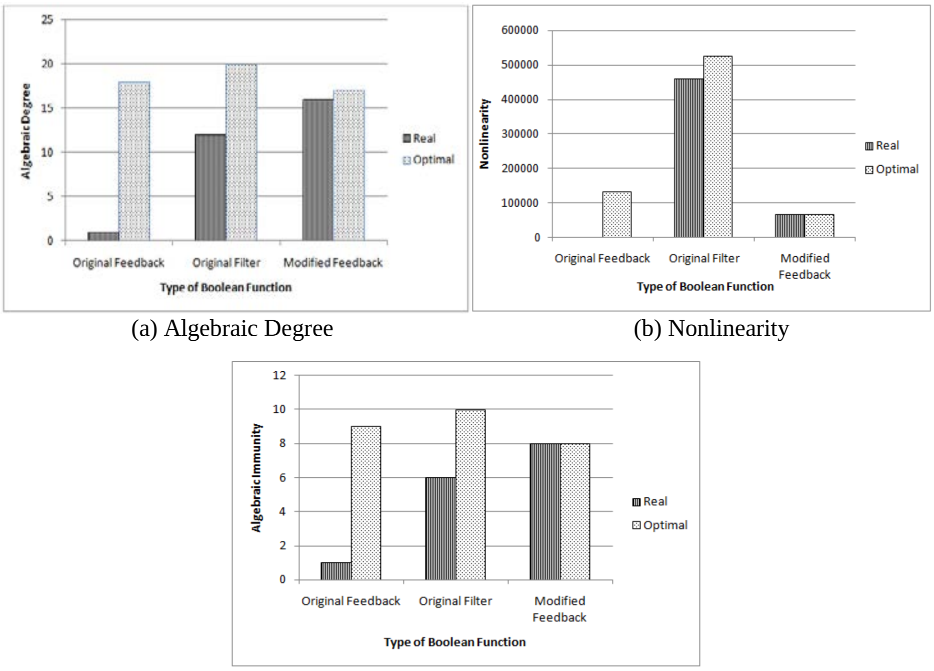
The degree of the primitive polynomials marks who setups are used as the input of the Boolean function in reverse order. This is because the least significant bit (LSB) in the Crypto-1 cryptosystem is on the left.

4.2 Population/Sampling

Generally, the size of a Mifare RFID tag is 2K, which offers two thousand and forty eight bytes, or 4K, and requires four thousand and ninety six bytes of data storage. The higher the amount of data that is saved in the tag, the more time is needed to encrypt. The RFID tag is usually used to save the ID of the owner. Indonesia has two main kinds of IDs: the *Kartu Tanda Penduduk*– KTP (citizen identity card), which requires about twelve bits, and the *Surat Ijin Mengemudi*– SIM (driving license), which requires about sixteen bits of storage data. Another commonly used ID is the student ID, in this example from Telkom University. The number of digits in this student ID was ten bits. Based on these three cases, three types of input messages with ten, twelve and sixteen bits will be tested.

4.3 Experiments

Three types of Boolean functions were used to construct the feedback shift register in this research. Figure 9 is the result of comparing the cryptographic properties among them.



(a) Algebraic Degree

(b) Nonlinearity

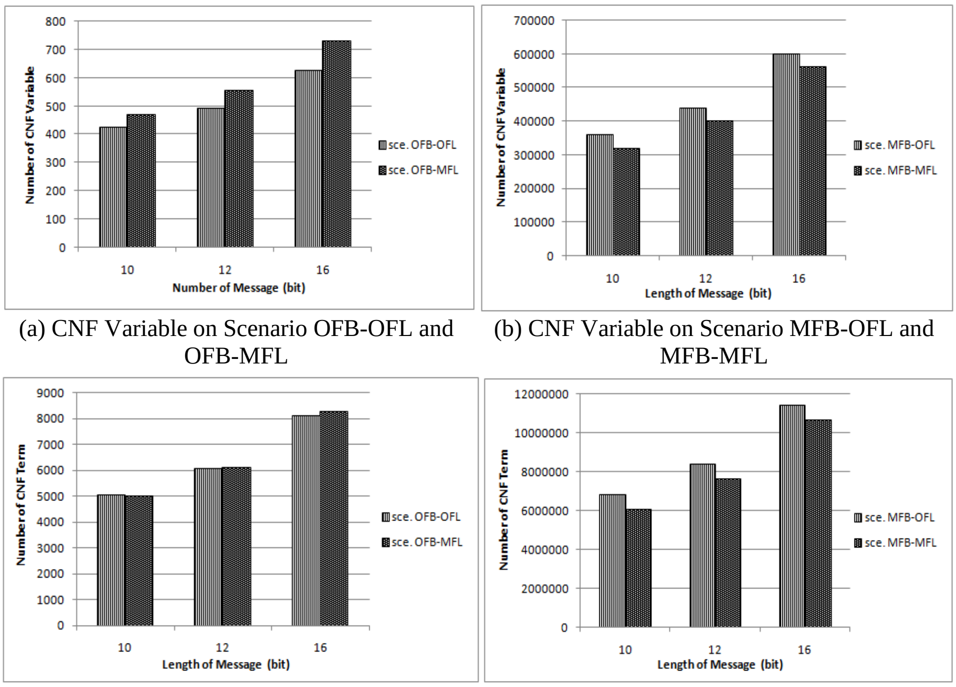
(c) Algebraic Immunity

Figure 9 Cryptographic properties comparison.

Based on the cryptographic properties calculation in Figure 9, it can be concluded that:

a. All of the Boolean functions fulfill the balancedness requirement. This means that the support functions fill half of their truth table.
b. The distance of algebraic degree, nonlinearity and algebraic immunity between the original feedback function and their own optimal value is about 94.44%, 100%, and 88.89%. These distances in all properties are the highest among these Boolean functions.
c. The distance of algebraic degree, nonlinearity and algebraic immunity between the modified feedback function and their own optimal value is about 5.89%, 0.16%, and zero. These distances in all properties are the lowest among these Boolean functions. The algebraic immunity property even has no distance to its optimal value.
d. The distance of algebraic degree, nonlinearity and algebraic immunity between the original filter function and their own optimal value is about 40%, 12.41%, and 40%.

Although the number of input variables in the modified feedback function is the lowest among these Boolean functions, it has the highest value of algebraic



(a) CNF Variable on Scenario OFB-OFL and OFB-MFL

(b) CNF Variable on Scenario MFB-OFL and MFB-MFL

(c) CNF Term on Scenario OFB-OFL and OFB-MFL

(d) CNF Term on Scenario MFB-OFL and MFB-MFL

Figure 10 SAT Solver attack preprocessing.

degree, nonlinearity and algebraic immunity. The cryptographic properties of the modified Boolean function are near to their optimal values. The algebraic immunity even reaches its optimal value.

These Boolean functions are used to encrypt the messages based on the scenarios in Section 4.1. There are four types of feedback shift registers based on the combination of the feedback and filter functions. In each scenario an algebraic attack was executed using an SAT solver. The initialization steps were not analyzed in the experiments; the attacks focus on the encryption process. An SAT solver attack consists of two main steps. First, it converts the ANF of the Boolean function into CNF. The results from our experiments can be seen in Figure 10. Based on Figure 10, it can be seen that the MFB-OFL scenario (the feedback shift register consists of the modified Boolean feedback function and the original filter function) has the highest number of CNF variables and terms. Furthermore, attack preprocessing in the MFB-OFL Scenario requires 1.2 GB to save the CNF file, which is the largest one.

The second step of an SAT solver attacks is finding the value of each literal in CNF using CryptoMiniSat. Both the MFB-OFL and MFB-MFL scenario, which use the modified Boolean feedback function, could not be attacked directly by CryptoMiniSat because of stack smashing. Stack smashing is a protection by the C programming language to detect buffer overflow errors. The number of clauses or terms was so high that it was hard to guess each literal or variable in these clauses. Furthermore, the stack as a data structure to store assigned literals reached its upper bound buffer limit because of some conflicting ANF terms. These terms must be removed so that the number of ANF parts can be increased.

Table 2 Conflicting terms in MFB-OFL scenario.

Iteration Number	Term Number	Term Value	Number of Revealed Variables	Percentage of Revealed Variables
1	3003	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{29}+x_{35}+x_{39}+x_{42}$	17268	4.786%
2	3015	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{27}+x_{29}+x_{39}$	17322	4.801%
3	3016	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{27}+x_{29}+x_{39}+x_{43}$	17322	4.801%
4	3017	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{27}+x_{29}+x_{39}+x_{42}$	17322	4.801%
5	3019	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{27}+x_{29}+x_{35}+x_{42}$	17322	4.801%
6	3023	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+x_{39}+x_{42}+x_{43}$	17322	4.801%
7	3024	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+ x_{35}+ x_{43}$	17322	4.801%
8	3025	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+ x_{35}+ x_{42}$	17322	4.801%
9	3026	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+x_{35}+x_{42}+x_{43}$	17322	4.801%
10	3027	$x_{12}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+ x_{35}+ x_{39}$	17322	4.801%

The higher the number of ANF parts that is attacked, the higher the number of CNF variables that will be revealed. A higher number of revealed CNF variables means a higher likelihood of recovering the secret key. The conflicting terms in a ten-bit input message can be seen in Tables 2 and 3.

The first iteration of Table 2 shows that removing ANF term number 3003, whose value is $x_{12} + x_{15} + x_{17} + x_{19} + x_{24} + x_{29} + x_{35} + x_{39} + x_{42}$, reveals 4.786% of the variables. This means that attacking 3002 ANF terms (exactly under the conflicting term that is 3003 minus 1) reveals the value of 17268 variables of the total of 360783 variables.

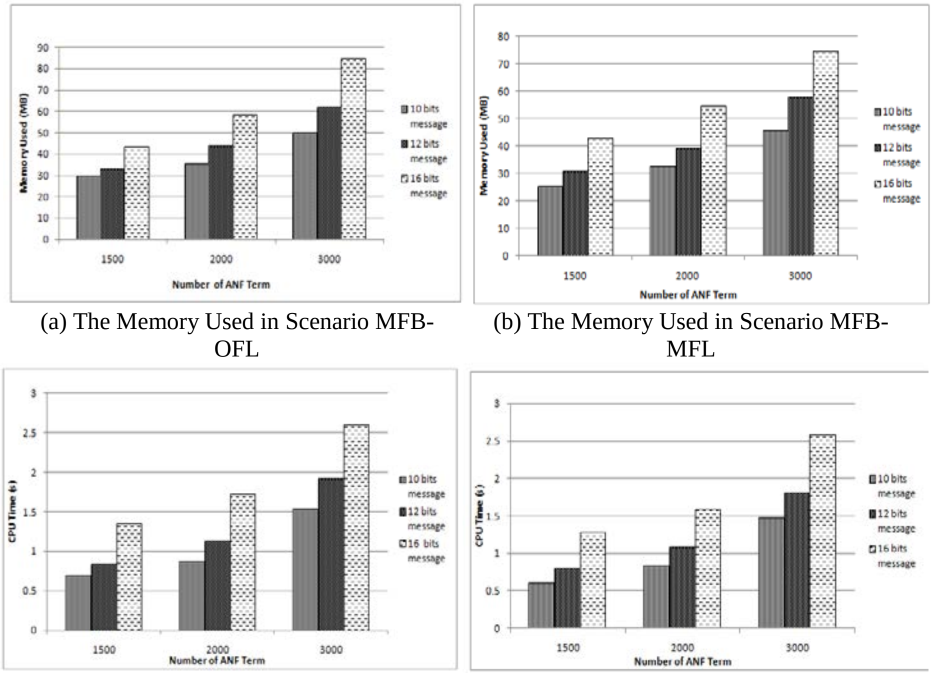
Table 3 Conflicting terms in MFB-MFL scenario.

Iteration Number	Term Number	Term Value	Number of Revealed Variables	Percentage of Revealed Variables
1	1527	$x_{14}+x_{15}+x_{17}+x_{19}+x_{24}+x_{25}+x_{35}+x_{39}+x_{42}$	10608	3.307%
2	1595	$x_{12}+x_{25}+x_{27}+x_{42}$	10944	3.412%
3	1597	$x_{12}+x_{25}+x_{27}+x_{39}$	10944	3.412%
4	1606	$x_{12}+x_{24}$	10992	3.427%
5	2009	$x_{12}+x_{17}+x_{19}+x_{24}+x_{29}+x_{39}$	12900	4.021%
6	2010	$x_{12}+x_{17}+x_{19}+x_{24}+x_{29}+x_{39}+x_{43}$	12900	4.021%
7	2021	$x_{12}+x_{17}+x_{19}+x_{24}+x_{27}+x_{35}+x_{39}+x_{42}$	12948	4.036%
8	2023	$x_{12}+x_{17}+x_{19}+x_{24}+x_{27}+x_{29}$	12948	4.036%
9	2033	$x_{12}+x_{17}+x_{19}+x_{24}+x_{25}+x_{43}$	12996	4.051%
10	2042	$x_{12}+x_{17}+x_{19}+x_{24}+x_{25}+x_{35}+x_{42}+x_{43}$	13036	4.064%

The first iteration of Table 3 shows that removing ANF term number 1527, whose value is $x_{14} + x_{15} + x_{17} + x_{19} + x_{24} + x_{25} + x_{35} + x_{39} + x_{42}$, reveals 3.307% of the variables. This means that attacking 1526 ANF terms (exactly under the conflicting term that is 1527 minus 1) reveals the value of 10608 variables of the total of 320782 variables.

Based on Tables 2 and 3, it can be concluded that the distance of iteration in the MFB-OFL scenario was lower than that in the MFB-MFL scenario, although the number of conflicting terms in the first iteration of the MFB-OFL scenario was higher than that in the MFB-MFL scenario. The MFB-MFL scenario produced conflicts starting from term number 1527, while the MFB-OFL scenario produced conflicts starting from term number 3003. Furthermore, the last five conflicting terms in the MFB-OFL scenario were consecutive terms and the number of revealed variables was constant after the second iteration. Based on these facts, it can be predicted that the number of revealed variables in the MFB-OFL scenario will remain constant but in the MFB-MFL scenario will increase gradually. The reason is because the number of input tap positions that are the same in the feedback and filter function is higher in the MFB-OFL scenario than in the MFB-MFL scenario. This means that the variance of variables in the MFB-OFL scenario is lower than in the MFB-MFL scenario. Based on this evidence, it can be concluded that the complexity needed to attack the MFB-OFL scenario at the first iteration is lower than for the MFB-MFL scenario.

The other alternative to attack the modified Boolean function is grouping the terms based on the area of the conflicted terms. Experiments were conducted in this study to find the amount of memory and CPU time needed for an attack among the total of 65133 ANF clauses in the MFB-OFL and MFB-MFL scenarios. The results can be seen in Figure 11. There were three groups of ANF terms consisting of 1500, 2500, and 3000 terms respectively. The CPU time needed was calculated from the average value of ten data. The amount of memory needed was stable, which means that the value was always the same, although the experiment was run more than once.



(a) The Memory Used in Scenario MFB-OFL

(b) The Memory Used in Scenario MFB-MFL

(c) The CPU Time in Scenario MFB-OFL

(d) The CPU Time in Scenario MFB-MFL

Figure 11 CryptoMiniSat complexity.

Based on Figure 11, it can be concluded that it was more complicated to attack the feedback shift register in the MFB-OFL scenario than in the MFB-MFL scenario because the amount of memory used in the MFB-OFL scenario was higher than in the MFB-MFL scenario. Since the use of memory depends on the number of CNF variables and terms in a scenario, it can be concluded that the amount of memory used in the MFB-OFL scenario was higher than in the MFB-MFL scenario because the CNF variables and terms in the MFB-OFL scenario were higher than in the MFB-MFL scenario. Furthermore, the CPU time in the MFB-OFL scenario was higher than in the MFB-MFL scenario because the number of CNF variables and terms in the MFB-OFL scenario were higher than in the MFB-MFL scenario. Moreover, more time was required to solve the higher number of equations.

4.4 Discussion

Based on the attacking processes in the MFB-OFL and MFB-MFL scenario, the drawbacks and benefits among these scenarios are as shown in Table 4.

Table 4 Comparison between MFB-OFL and MFB-MFL scenario.

Scenario	Drawbacks	Benefits
MFB-OFL	The first attack iteration gives a high number of revealed variables	Memory & CPU time for attacking process are high
MFB-MFL	Memory & CPU time needed for attacking process are low	The first attack iteration gives a low number of revealed variables

If computer resources are considered, then the MFB-OFL scenario, which consists of the modified feedback function and the original filter function of Crypto-1, is the best choice for the feedback shift register since the MFB-OFL scenario needs more CPU time. This scenario needs about 2.596 seconds and 84.50 MB of memory to attack 3000 from the total of 65133 ANF terms. However, this is just 4.6% of the total attack processing. In addition, the time required to attack is higher than to encrypt the message, i.e. about 2.315 seconds.

If the number of revealed variables is considered, then the MFB-MFL scenario, which consists of the modified feedback function and the modified filter function of Crypto-1, is the best choice for the feedback shift register since this scenario has a lower number of revealed variables than the MFB-OFL scenario. The MFB-MFL scenario has a lower number of revealed variables in the first iteration than the MFB-OFL scenario and still has a lower number of revealed variables even if the number of revealed variables in the next iteration is increased.

5 Conclusion

This paper proposed a modified Boolean feedback function to protect the Crypto-1 encryption algorithm against SAT solver attacks. The modification was developed for overcoming weaknesses of Crypto-1, especially the random number quality. For this reason, it uses Wang's construction because this has cryptographic properties near the optimal values. Moreover, it still has the possibility to be implemented in embedded systems.

Based on the experiments conducted in this research, it was shown that the modified Boolean feedback function had the best cryptographic properties although the balancedness property among all functions is the same. Moreover, the algebraic immunity in the modified Boolean feedback function reaches optimal value and can be categorized as a perfect algebraic immune system.

The modified Crypto-1 consisting of the modified feedback function and the original filter function is the most complicated combination to attack. Therefore, this feedback shift register is appropriate to be selected if computer resources are considered. The modified Crypto-1 consisting of the modified feedback function and the modified filter function has the lowest percentage of revealed variables so that it is appropriate to be selected as the best feedback shift register if the number of revealed variables is considered. Finally, it can be concluded that the modified Crypto-1 provides stronger protection against SAT solver attacks than the original Crypto-1. However, implementation of the modified Crypto-1 is still an open problem. It needs to be further analyzed so that the complexity of the implementation of the modified Boolean feedback function can be reduced.

Acknowledgements

We would like to express our gratitude to the director and all lecturers of Telkom University, especially at the Master Training Program of the Graduate School, and friends for their moral contributions, given to me either directly or indirectly throughout my study and the development of my project.

References

[1] Garcia, F.D.,van Rossum, P., Verdult, R. & Schreur, R.W., *Wirelessly Pickpocketing a Mifare Classic Card*, Proceedings of the 30th IEEE Symposium on Security and Privacy, pp. 3-15, Washington, DC, USA, 2009.

[2] Courtois, N.T. & Nohl, K., *Algebraic Attacks on the Crypto-1 Stream Cipher in Mifare Classic and Oyster Cards*, Cryptology ePrint Archive, Report 2008/166,2008.

[3] Garcia, F.D., Gans, G.K., Muijrers, R., Rossum, P., Verdult, R., Schreur, R.W. & Jacobs, B., *Dismantling Mifare Classic*, Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security, ESORICS ’08, pp. 97-114, Berlin, Heidelberg, 2008.

[4] Nohl, K., *Cryptanalysis of Crypto-1*, Computer Science Department University of Virginia, White Paper, 2008.

[5] Soos, M., *Privacy-preserving Security Protocols for RFIDs*, Master’s Thesis, Institut Polytechnique de Grenoble, Grenoble, France, 6 October 2009.

[6] Liu, M., Zhang, Y. & Lin,D., *Perfect Algebraic Immune Functions*, Advances in Cryptology–ASIACRYPT 2012, pp. 172-189, Springer, 2012.

[7] Courtois, N.T. & Meier, W., *Algebraic Attacks on Stream Ciphers with Linear Feedback*, Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques, EUROCRYPT ’03, pp. 345-359, Berlin, Heidelberg, 2003.

[8] Eén, N. & Sorensson, N., *An Extensible Sat-Solvers*, Theory and Applications of Satisfiability Testing, **2919**, pp. 502-518, Springer Berlin Heidelberg, 2004.

[9] Bard, G.V., Courtois, N.T. & Jefferson, C., *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials Over Gf(*2*) Via Sat-Solvers*, Cryptology ePrint Archive, Report 2007/024, 2007.

[10] Nohl, K., *Disclosing Secret Algorithms from Hardware*, Black Hat Japan, 2008.

[11] Courtois, N.T., O’Neil, S. & Quisquater, J.-J., *Practical Algebraic Attacks on the Hitag2 Stream Cipher*, Proceedings of the 12thInternational Conference, ISC 2009, **5735**, pp.167-176, Pisa, Italy, September 7-9, 2009.

[12] Soos, M., Nohl, K. & Castelluccia, C., *Extending Sat Solvers to Cryptographic Problems*, Proceedings of the 12th International Conference, SAT 2009, pp. 244-257, Swansea, UK, June 30-July 3, 2009.

[13] Wang, Q., Peng, J., Kan, H. & Xue, X., *Constructions of Cryptographically Significant Boolean Functions Using Primitive Polynomials*, Information Theory, IEEE Transactions on, **56**(6), pp.3048-3053, 2010.

[14] Carlet, C., *Comments on “Constructions of Cryptographically Significant Boolean Functions Using Primitive Polynomials”*, Information Theory, IEEE Transactions on Information Theory, **57**(7), pp.4852-4853, 2011.

[15] Fischer, S., *Analysis of Lightweight Stream Ciphers*, Master’s Thesis, EcolePolytechniqueFederale De Lausanne, Lausanne, Switzerland, April 2008.

[16] Cormen, T.H., Leiserson, C.E., Rivest, R.L. & Stein, C., *Introduction to Algorithms*, (3rded.), MIT press Cambridge, 2009.

[17] Bard, G.V., *Algebraic Cryptanalysis*, Springer Publishing Company, United States, 2009.

[18] Soos, M., *Grain of Salt – An Automated Way to Test Stream Ciphers Through SAT Solvers*, Workshop on Tools for Cryptanalysis, June, 2010.

[19] Carlet, C., *Boolean Functions for Cryptography and Error Correcting Codes*, Boolean Models and Methods in Mathematics, Computer Science, and Engineering, **2**, pp. 257, 2010.

[20] Townsend,W.J. & Thornton, M.A., *Walsh Spectrum Computations using Cayley Graphs*, Spectrum, **15**(11), p.5, 2001.