Advances in Computer Science Research, volume 96
Proceedings of the International Conference on Mathematics, Geometry, Statistics, and Computation (IC-MaGeStiC 2021)

DOPE: MDC-2 Scheme Using PRESENT

Anjeli Lutfiani*, Bety Hayat Susanti

Politeknik Siber dan Sandi Negara
Jalan Haji Usa Raya, Ciseeng, Bogor, Indonesia, 16120
**Corresponding author. Email: anjeli.lutfiani@student.poltekssn.ac.id*

ABSTRACT
Modification Detection Code (MDC) as an unkeyed hash function is designed to provide data integrity. Manipulation Detection Codes (MDC-2) is one of double-length ($2n$-bit) hash-values that requires 2 block cipher operations per block of hash input where the output size of the hash function is twice the size of the block cipher. Constructing hash function from block ciphers as in MDC-2 is expected to produce a hashing algorithm that has the same efficiency and properties that are following its use as a block cipher. In this paper, we construct a Double-length Matyas-Meyer-Oseas based on PRESENT (DOPE) hash algorithm, that implements PRESENT as a lightweight block cipher on the MDC-2 scheme. PRESENT is used as the primary compression function with 64-bit block message and 80-bit key as the inputs. To analyze the performance and resistance of DOPE against collision, a test is conducted using Yuval's Birthday Attack. It generates minor modification input of 2^{32} on extreme input pairs with uniform values and input pairs with random values, and it is proven to be collision-resistant.

Keywords: *Hash function, Lightweight block cipher, PRESENT, MDC-2, Yuval's birthday attack.*

1. INTRODUCTION

A hash function is a function that maps an arbitrary length input bit string to a fixed-length output, with its basic nature as an easy-to-compute compression function [1]. The output produced by the hash function is called hash code, hash result, hash value, or hash only [1]. One of the purposes of hash functions is to provide data integrity services. To be secure, there are three main properties that the hash function hash to accomplish i.e., preimage resistance, second preimage resistance, and collision resistance [2].

Based on the use of keys, the hash function is divided into unkeyed hash function and keyed hash function. One form of an unkeyed hash function is a block cipher-based hash function which is divided into hash functions that produce hash values of single length (n-bit) and double-length ($2n$-bit). The basic idea of building a block cipher-based hash function is that if the implementation of a block cipher is said to be efficient, then the use of block ciphers as the main component is expected to provide the latter functionality at a small additional cost [1].

Based on [2], the construction of a single-length hash function based on block ciphers is known to be not collision-resistant. Then the construction of double-length hash function is a solution to increase hash function resistance against collisions [3]. It indicates that for an n-bit block that produces a $2n$-bit hash value, the requisite work to find the collision in the hash function is expected to be 2^n.

However, in his thesis, Thomsen found that there were several implications for some properties of the MDC-2 scheme. The research proves that the security level in the general construction of MDC-2 is less than optimal because the scheme is vulnerable to collision attacks and preimage attacks [3].

In 2008, A. Bogdanov et al. [4] conducted research on the construction of a block cipher-based hash function. In this research, the construction of single-length hash functions and double-length hash functions using AES and PRESENT as the main component. In conclusion, among all variants of block cipher-based hash function construction, H-PRESENT-128 is the best candidate equivalent to AES and has a value of 50% smaller than MD5 with the GE area value is 4000 GE. This algorithm also only requires 20 to 30 times fewer clock cycles than AES and MD5 [4].

In 2014, there was a comparative analysis of double-length and single-length hash construction using the Matyas-Meyer-Oseas scheme based on AES-192 or

AES-256. Analysis was carried out on the collision resistance property, preimage resistance, and other properties. The results showed that the double-length hash scheme had better robustness [5]. Furthermore, Bos *et al*. showed that the hash function construction with the Davies Meyer scheme and others can provide advantages in exploiting hardware features [6].

Based on these matters, the construction of MDC-2 will be modified using Matyas-Meyer-Oseas based on PRESENT, namely DOPE (Double-length Matyas-Meyer-Oseas based on PRESENT). This is done considering that PRESENT as a standard lightweight block cipher algorithm in ISO/IEC 29192-2:2019 [7] is expected to meet implementation needs in a limited environment.

In [8] and [9], Yuval's birthday attack was carried out with the goal to find the collision. The test works by conducting 120 experiments of extreme input and pseudorandom input from the implementation of the Simplified AES (S-AES) algorithm as a compression function on MMO, DM, and MP schemes, however, the three schemes are not resistant to the collision [8]. Besides, Yuval's birthday attack on the Simplified Chaos Hash Algorithm-1 (SCHA-1) shows collision modus of 1 which occurs 40 times in 120 trials [9]. We conducted Yuval's Birthday Attack to find the number of collisions for the modified scheme. Based on [2], it is easier to find collisions for a one-way hash function than to find preimages or second preimages of specific hash values.

2. HASH FUNCTION

In general, hash functions are divided into two types, namely unkeyed hash functions and keyed hash functions. An unkeyed hash function is a hash function that works using a single input parameter, i.e message. While keyed hash function is a hash function that works by using two input parameters, specifically message and secret key.

One of the subclasses of unkeyed hash functions is MDC (Modification Detection Code) which aims to provide data integrity service that is required by certain applications [2]. While one of the subclasses of keyed hash function is MAC (Modification Authentication Code) which facilitates data authentication service [2].

Based on the compression function operations applied, MDC is categorized into block cipher-based hash functions, customized hash functions, and modular arithmetic-based hash functions [2].

The following in Figure 1 are three well-known schemes of single-length hash function from the twelve best schemes [1] wherein E can be any of the compression function operations previously mentioned.

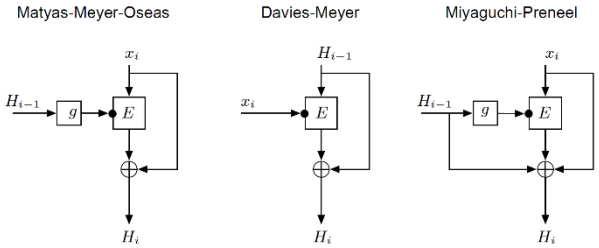

Figure 1 Three well-known single-length schemes [2].

Those algorithms are proven to be secure under a suitable black-box model, assuming E satisfies the required randomness property and no internal properties or weaknesses of E are exploited in the attack, i.e., finding preimages and collisions requires 2^n and $2^{n/2}$ operations [2].

However, because they are single-length, none of the three is collision-resistant for block-based ciphers with relatively small lengths [2]. In this case, double-length is intended to produce hash-codes with a length of $2n$ bits. It uses an n-bit block cipher as the main component and yielding $2n$-bit string as output [10]. One form of a well-known double-length hash function is MDC-2.

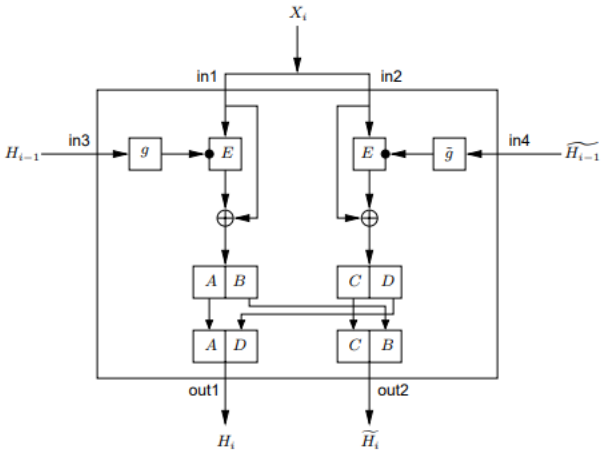

Figure 2 MDC-2 Scheme [2].

MDC-2 is a manipulation detection code that works with input parameters of 2 block cipher operations per block input hash and generates 128-bit hash as the output [2]. This scheme as shown in Figure 2 uses Matyas-Meyer-Oseas scheme based on the DES algorithm by adding other properties to produce an algorithm that is resistant to collisions compared to single-length Matyas-Meyer-Oseas based on DES.

3. YUVAL'S BIRTHDAY ATTACK

The three main properties of the hash function are preimage resistance, second preimage resistance, and collision resistance. By definition in [11], preimage resistance i.e., for a hash value h, it is impossible to find an input message m that satisfies $h = hash\,(m)$. Second preimage resistance, that is, for an input message value of m_1, it is impossible to find an input message m_2 that satisfies $hash\,(m_1) = hash\,(m_2)$ where m_1 is not equal to m_2. Collision resistance, i.e., it is not possible to find input messages m_1 and m_2 that have the same hash value.

A collision attack is an attempt to find two input messages m_1 and m_2 that have the same hash value. It takes $2^{n}/_{2}$ tries to find a pair of messages m_1 and m_2 that has the same hash value, where n represents the length of the hash value.

Yuval's Birthday Attack is one of the collision attacks based on the birthday paradox theory. For example, in a group that contains 23 people, then the probability that there are at least two people who have a birthday on the same day is more than 0.5. In [12] it is explained that Yuval's Birthday Attack can be used to minimize experiments in finding collisions, or it can be said that this attack is useful for making it easier to find collisions in a hash function.

This attack algorithm requires n-bit original message x_1 and n-bit fake message x_2 as the inputs with outputs x_1' and x_2' as a result of minor modification of x_1 and x_2 with $h(x_1') = h(x_2')$. This attack generates $2^{n/2}$ minor modifications of x_1' and hash it. Then also generates $2^{n/2}$ minor modifications of $x_2{}'$ and hash it. Finally, compare the results and check whether there is a collision between the two hash values [2].

4. ANALYSIS

4.1. DOPE: MDC-2 based on PRESENT

The DOPE (Double-length Matyas-Meyer-Oseas based on PRESENT) algorithm as shown in Figure 3 is a modified MDC-2 scheme based on the PRESENT algorithm as an alternative to the DES algorithm. DOPE processes 128-bit plaintext blocks or multiples with an 80-bit secret key obtained from the output function g with IV input (padding if necessary). Both the plaintext block and the secret key are then processed in the PRESENT algorithm as a function E. The output of DOPE is a 128-bit fixed-length hash value that has been previously linearly transformed.

The Matyas-Meyer-Oseas scheme is one of the three well-known schemes based on [1]. MDC-2 uses one of the implementations of the Matyas-Meyer-Oseas scheme based on the DES algorithm by adding other properties to produce an algorithm that is resistant to collisions compared to single-length Matyas-Meyer-Oseas based on DES.

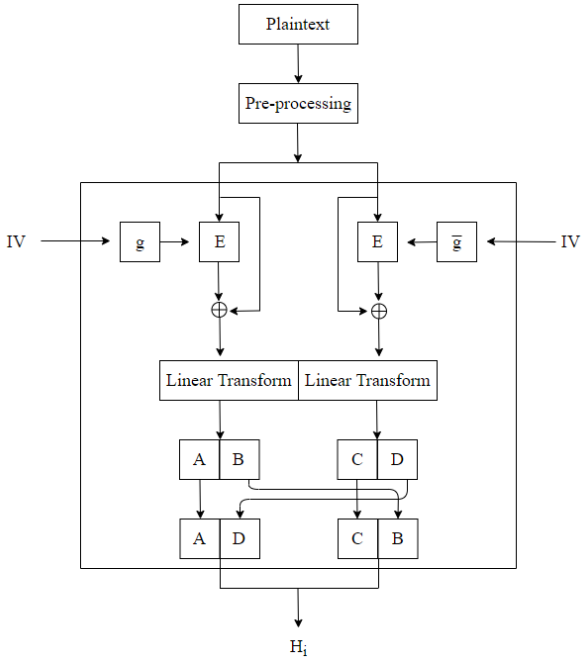

Figure 3 DOPE Algorithm Scheme.

4.1.1. Pre-processing Phase

In the original description of MDC-2 [2], padding may be required to meet the bit length restriction on the input. As defined before, DOPE algorithm needs an input of length 128-bit or multiple. The addition of the pre-processing phase property to the DOPE algorithm intends to set input messages that do not contain 128-bits or multiples. This input next will be padded by 0*.

For example, with input message 11111111 22222222 33333333 44 (in hexadecimal) that consist only of 104 bits, through the pre-processing phase it will be 128 bits, i.e., 11111111 22222222 33333333 44000000 (in hexadecimal).

The padding process in DOPE is one of the simple process in the "Handbook of Applied Cryptography". The ambiguous nature of this padding method is expected to be able to minimize the possibility of attackers knowing the message input used [2].

4.1.2. Permutation Function

The input IV in the original description of MDC-2 [2] H_{i-1} and $\overline{H_{i-1}}$ were fixed. The default set of them were 52525252 52525252 (in hexadecimal) and 25252525 25252525 (in hexadecimal). This holds two implications that all known weak and semi-weak keys of DES were decreed out and both of them were always different [3].

The existence of g and $\bar{g}$ functions as permutation functions in DOPE is intended to obtain the diffusion of key inputs before entering the key scheduling process.

The permutation functions g and $\bar{g}$ in the DOPE algorithm treat 64-bit input as four 16-bit nibbles.

$$g\ (IV) : IV_A \mid IV_B \mid 00000000\ 00000001 \mid IV_C \mid IV_D$$

$$\bar{g}\ (IV) : IV_A \mid IV_B \mid 10000000\ 00000000 \mid IV_C \mid IV_D$$

The Matyas-Meyer-Oseas structure which makes input IV, not as input is feared to be an entrance for attackers if the permutation process is not carried out beforehand.The permutation function is also different between the left block and the right block. This is done so that if the IV used as input has the same value, the result of the permutation which will be the key input will not produce the same value.

4.1.3. Encryption Function

The construction of MDC-2 arises with the matter that none of the single-length hash functions based on block ciphers with relatively small lengths is collision-resistant [2]. It was formerly designated using DES as the underlying block cipher.

Following the idea of its manufacture, MDC-2 is expected to have good resistance to the collision, that given an ideal block cipher, MDC-2 will present a collision-resistant hash function. However, [3] proves that the security level in the general construction of MDC-2 is less than optimal because the scheme is vulnerable to collision attacks and preimage attacks

The use of PRESENT as the basis for DOPE is based on the fact that SPN scheme which is the basis of the PRESENT algorithm hopefully to be able to perform computations faster than the Feistel scheme as in DES.

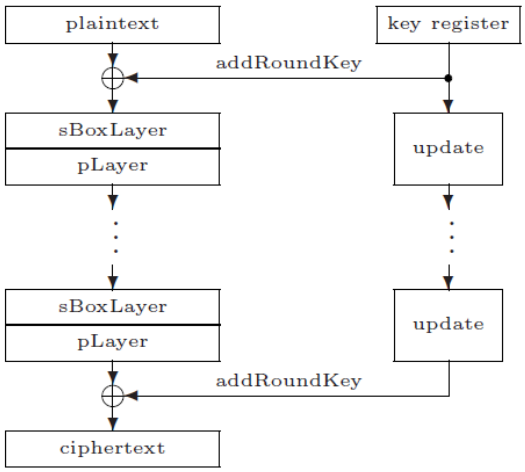

Figure 4 PRESENT Algorithm Scheme [7].

PRESENT algorithm [7] as shown in Figure 4 is a standard algorithm of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) which was created in 2007. This algorithm works by mapping 64-bit plaintext blocks into 64-bit ciphertext with 80 or 128-bit secret keys. PRESENT is proposed as an optimization of the hardware implementation and is claimed to have a Gate Equivalent (GE) size of 2.5 times smaller than the AES algorithm [7].

PRESENT is an SPN algorithm that operates in 31 rounds, with an 80 or 128-bit secret key that is operated into the KeySchedule process to generate 32 subkeys. The SPN structure in the PRESENT consists of XOR operations on AddRoundKey, non-linear substitution operations on SboxLayer, and linear permutations on Player. The 32nd round is the post-whitening stage, which contains the XOR operation with the 32nd subkey

4.1.3.1. AddRoundKey

Given the round key K_r for $1 \leq r \leq 32$ along with the last 64-bit state, addRoundKey works with the operation,

$$b_i \rightarrow b_i \oplus k_i^r \qquad ,0 \leq i \leq 63.$$

4.1.3.2. SBoxLayer

PRESENT algorithm use s-box $S : \mathbb{F}_2^4 \rightarrow \mathbb{F}_2^4$ where for the 16 four-bit word w_{15} ... w_0,, we know the value $w_i = b_{4*i+3}||b_{4*i+2}||b_{4*i+1}||b_{4*i}$ with $0 \leq i \leq 15$. The following Table 1 is a PRESENT s-box table.

Table 1. Table S-Box [7]

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

4.1.3.3. PLayer

The permutation layer operation works by mapping each bit of the last state as shown in Table 2.

Table 2. Table Permutation [7]

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P (i)	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
P (i)	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
P (i)	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
P (i)	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

4.1.3.4. KeySchedule

Given a secret key K which is represented as $k_{79}k_{78}$ ... k_0. Round key K_i in round i generate with

$$K_i = k_{63}k_{62} \ldots k_0 = k_{79}k_{78} \ldots k_{16}$$

After extraction of the round key K_i , the register $K = k_{79}k_{78} \ldots k_0$ is updated as shown in Figure 5.

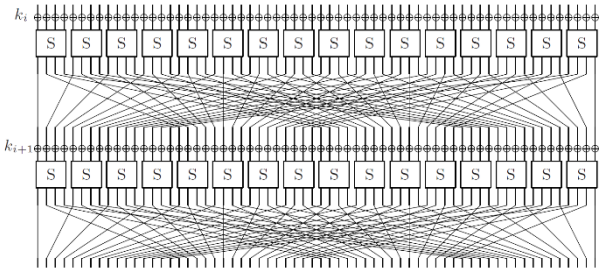

Figure 5 S/P Network PRESENT Algorithm [7].

1. $[k_{79}k_{78} \ldots k_1k_0] = [k_{18}k_{17} \ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus round\_counter$

4.1.4. Linear Transform

A linear transformation is carried out after the encryption results from the E function are obtained. DOPE will be divided into four 16-bit nibbles. Furthermore, the four nibbles will go through a linear transformation phase that XOR the four nibbles with nibbles that have been circular left-shifted of 6 bits and 10 bits.

$$L : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$$

$$L\,(c) : c \oplus (c \ll 6) \oplus (c \ll 10)$$

In contrast to MDC-2, DOPE not only performs permutations of the encryption results but also performs linear transformations. Linear transformations in the form of rotation and xor-ing have the same goal to obtain diffusion. In DOPE, the rotation values are 6 and 10, with the hope that the rotation will not cause the bit to return to its initial position [13].

4.2. Collision of DOPE with Extreme Input

We use extreme hex values from 00000000 00000000 00000000 00000000 to ffffffff ffffffff ffffffff ffffffff. Experiments on collision values are applied to all possible pairs of extreme value inputs, which are 120 combinations of minor modified input pairs. Where each input pair takes a sample of minor modifications by changing the last 32-bit of the message as much as 2^{32} values.

First, generate the hash value of input 0000 ... 0000 by changing the last 32-bit values, and then store it. Second, generate the hash value of input 1111 ... 1111 by changing the last 32-bit values, and then store it. All values that have been stored are then compared and examined whether there are similar values.

Based on the results of the comparison, it is known that there is no hash value of the entire input comparison. This indicates that the test on the first pair of inputs does not find any collisions. Next, do the same for the 120 possible input pairs. Based on the test results as in Table 3, it can be concluded that there is no collision in the use of extreme input pairs.

Table 3. Collision results of extreme input pairs

Pairs of the message input		**Number of Collisions**
0000 … 0000	1111 … 1111	-
0000 … 0000	2222 … 2222	-
0000 … 0000	3333 … 3333	-
0000 … 0000	4444 … 4444	-
0000 … 0000	5555 … 5555	-
0000 … 0000	6666 … 6666	-
0000 … 0000	7777 … 7777	-
0000 … 0000	8888 … 8888	-
0000 … 0000	9999 … 9999	-
0000 … 0000	aaaa … aaaa	-
0000 … 0000	bbbb … bbbb	-
0000 … 0000	cccc … cccc	-
…	…	-
…	…	-
…	…	-
eeee … eeee	bbbb … bbbb	
eeee … eeee	cccc … cccc	
eeee … eeee	dddd … dddd	-
eeee … eeee	ffff … ffff	-

4.3. Collision of DOPE with Pseudorandom Input

Experiments on collision values with pseudorandom inputs are applied to all possible pairs in the same way as experiments using extreme inputs. The pseudorandom input value is generated using a random number generator in the C++ programming language based on time on Windows 10 Pro OS. Here is Table 4 consists of the results of 16 pseudorandom inputs that will then be used as 120 combinations of input pairs.

Table 4. Pseudorandom input

Pseudorandom input
9b787591 c435f237 3661c5b6 1a7713a3
bc2280f4 cc65d6a4 3e851523 2bfa8d04
1f62b52d a9dd789a bd71c3ba 75d8f820
4cae8eb4 e2e44f8f 3bcf49e3 5fb9cd81
cb8e6320 1f21d4fd 4fd52e45 7204a3ad
057cda0a b92d9d6a 514cda39 3543f30d
91fe7dbb 3740145e f97ce657 10fc4338
e88e94d8 2121c153 a00dd907 ab980e98
81b2e5cc f03a3bcf dc661be1 6fbfb9b3
f4e3ba2e 3be1ea4c 1800256c c3b9fe22
a9a8bb65 5ad04630 d984aef1 402e144d
197b4efa e57cd834 a957ee39 8c95c3bc
dbe2ed75 654027af 0ff49d99 d16764d7
67562f5e 41c3bb2b 64e2149c e73c7d45
366f8d1d 127cfc2e 4fa8eab9 146c897f
df757e39 4fd47311 38bf9877 e29f1ded

The same as the process carried out on the search for the collision of DOPE using extreme input, each pseudorandom input pair takes a sample of minor modifications by changing the last 32-bit of the message as much as 2^{32} values and comparing it. The test of using pseudorandom input pairs is carried out according to the following process.

First, generate the hash value of input 9b787591 c435f237 3661c5b6 1a7713a3 by changing the last 32-bit values, and then store. Second, generate the hash value of input bc2280f4 cc65d6a4 3e851523 2bfa8d04 by changing the last 32-bit values, and then store it. All values that have been stored are then compared and examined whether there are similar values.

Based on the results of the comparison, it is known that there is no hash value of the entire input comparison. This indicates that the test on the first pair of inputs does not find any collisions. Next, do the same for the 120 possible pseudorandom input pairs. Based on 120×2^{32} tries as shown in Table 5, none of the collision was found from the hash values produced

Table 5. Collision results of pseudorandom input pairs

Pairs of the message input		Number of Collisions
9b78 … 13a3	bc22 … 8d04	-
9b78 … 13a3	1f62 … f820	-
9b78 … 13a3	4cae … cd81	-
9b78 … 13a3	cb8e … a3ad	-
9b78 … 13a3	057c … f30d	-
9b78 … 13a3	91fe … 4338	-
9b78 … 13a3	e88e … 0e98	-
9b78 … 13a3	81b2 … b9b3	-
9b78 … 13a3	f4e3 … fe22	-
9b78 … 13a3	a9a8 … 144d	-
9b78 … 13a3	197b … c3bc	-
9b78 … 13a3	dbe2 … 64d7	-
…	…	-
…	…	-
366f … 897f	6756 … 7d45	-
366f … 897f	df75 … 1ded	-

5. CONCLUSION

The modification of MDC-2 using Matyas-Meyer-Oseas based on PRESENT, namely DOPE (Double-length Matyas-Meyer-Oseas based on PRESENT) is done. Modification is done by adding some additional properties such as pre-processing phase, permutation function, encryption function or PRESENT algorithm, and linear transform. Based on the results of Yuval's Birthday Attack with 120 extreme input pairs and 120 pseudorandom input pairs, none of the collisions from the minor modified input hash were found. This attack was done by changing the last 32-bit of input as much as 2^{32} values. We suggest future work to do Yuval's Birthday Attack on this scheme with 2 block input and other variant attack on trying to find collision on a larger input range.

REFERENCES

[1] L. R. Knudsen and M. J. , The Block Cipher Companion, Springer-Verlag Berlin Heidelberg, 2011.

[2] A. Menezes, P. v. Oorschot and S. Vanstone, Handbook of Applied Cryptography, USA: CRC Press, Inc, 1996.

[3] S. S. Thomsen, Cryptographic Hash Functions, 2009.

[4] A. Bogdanov et al, Hash Functions and RFID Tags: Mind The Gap, Springer-Verlag Berlin Heidelberg, 2008.

[5] S. Hirose and H. Kuwakado, A Block-Cipher-Based Hash Function Using an MMO-Type Double-Block Compression Function, Springer-Verlag Berlin Heidelberg, 2014.

[6] J. Bos, O. Özen and M. Stam, Efficient hashing using the AES instruction set, Springer-Verlag Berlin Heidelberg, 2011.

[7] A. Bogdanov et al, PRESENT: An Ultra-Lightweight Block Cipher, Springer-Verlag Berlin Heidelberg, 2007.

[8] E. Sabarina, B. Hayat Susanti and A. Winarno, Implementation Analysis of Simplified AES (S-AES) Algorithm on Matyas-Meyer-Oseas (MMO), Davies-Meyer (DM), and Miyaguchi-Preneel (MP) Schemes using Yuval's Birthday Attack, Information Systems International Conference (ISICO), 2013.

[9] A. Admi and B. Hayat Susanti, Studies on Simplified Chaos Hash Algorithm-1 (SCHA-1) using Yuval's Birthday Attack, International Conference on Mathematics, Statistics and their Applications (ICMSA), 2012.

[10] Z. Zolfaghari, H. Asadollahi and N. Bagheri, Multicollision Attack on a Recently Proposed Hash Function vMDC-2, Journal of Computing and Security (JComSec), 2016.

[11] F. Mendel, Analysis of Cryptographic Hash Functions, Austria: Graz University of Technology, 2010.

[12] K. Inayah and B. Estuwira, Collision Resistance Fungsi Hash Berbasis Block Cipher Dengan Menggunakan Algoritma Mini-AES, Seminar Nasional Sistem Informasi Indonesia, 2013.

[13] D. Engels, X. Fan, G. Gong, H. Hu and E. Smith, Hummingbird: Ultra-lightweight cryptography for resource-constrained devices, Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics);6054 LNCS:3–18., 2010.