Second Preimage Attack Method on Various MAC Constructions and Its Application with AES-128

Alfonso Brolin Sihite
Sekolah Tinggi Sandi Negara
Bogor, Indonesia
alfonso.brolin.s.17@gmail.com

Bety Hayat Susanti
Sekolah Tinggi Sandi Negara
Bogor, Indonesia
beth.susanti@gmail.com

***Abstract*—A Message Authentication Codes (MAC) can be constructed based on a block cipher algorithm. CBC-MAC, EMAC, ECBC-MAC, FCBC-MAC, XCBC-MAC, RMAC, TMAC, OMAC and CMAC are some of MAC constructions that used in the hash function. In this paper, we propose a method of second preimage attack that utilizes the concept of existential forgery on CBC-MAC and it can be used on all of that various MAC construction. We apply the method to find second preimage on that various MAC constructions which uses AES-128 block cipher algorithm in its basic construction for the proof that the method is working. The results show that with the modifications as many as 2^{20} for each sample, we can obtain the second preimages easily as many as 2^{20} in that various MAC constructions.**

Keywords*— AES; CBC-MAC; CMAC; EMAC; ECBC-MAC; existential forgery; FCBC-MAC; hash function; OMAC; second preimage attack; RMAC; TMAC; XCBC-MAC

I. Introduction

A hash function takes a string of the arbitrary length, then maps it to a fixed output length as a hash value [11]. Hash function can be used as error detection by attaching the hash value with the message during a transmission. The error will be detected if the hash value of the received message was not the same as the hash value that attached [10].

Based on its functions, hash functions are categorized into a Modification Detection Codes (without a key) and Message Authentication Codes (with a key). MAC can be designed by defining a new mode of operation for existing primitive [4]. Some of them are the CBC-MAC, EMAC, etc.

Advanced Encryption Standard (AES) is a block cipher standard algorithm which specified by the National Institute of Standards and Technology (NIST) in 2001 [8]. As a block cipher algorithm, AES has been proved to have good security and easy to implement [4]. AES process data blocks of 128 bits using varied secret key length, i.e. 128 bit, 192 bits, and 256 bits. AES that uses 128 bit key referred as AES-128 [8]. In this research, we use AES-128 as the block cipher in various MAC constructions i.e. Cipher Block Chaining Message Authentication Code (CBC-MAC), Encrypted Message Authentication Code (EMAC), ECBC-MAC, FCBC-MAC, XCBC-MAC, Randomized Message Authentication Code (RMAC), Two-Key CBC-MAC (TMAC), One-Key CBC-MAC (OMAC) and Cipher-Based Message Authentication Code (CMAC).

To be a good hash function, it must satisfy three main requirements, which are preimage resistance, second preimage resistance, and collision resistance [7]. Second preimage resistance is a condition where if given value x and the hash value H which satisfy $h(x) = H$, then it is computationally difficult to find a message $x' \neq x$ where $h(x) = h(x')$[1]. In 2009, Jia has proposed second preimage attack to CBC-likes MACs [5]. In this research, we propose a second preimage attack method which utilizes the concept of existential forgery on CBC-MAC that more efficient and effective than Jia's method. The method will be applied to various MAC constructions to prove that the method is working. The goal of this research is to know the second preimage attack method that more efficient than Jia's method which can be used to do second preimage attack on various MAC constructions based on block cipher algorithm.

II. Theoritical Background

A. Hash Function

A function that takes a string of the arbitrary length, then maps it to a fixed output length as a hash value is called as a hash function [11]. The basic idea of a cryptographic hash function is the hash value as an image representation (digital fingerprint, imprint, and message digest) from an input string and can be used only if the hash value can be uniquely identified with an input string [1]. A hash function must satisfy the following three properties [1]:

1) *Preimage resistance*
   Given a hash value y, then it is computationally difficult to find m which satisfy $F(m) = y$.
2) *Second preimage resistance*
   Given $(m, F(m))$, it is computationally difficult to find m', with $m' \neq m$, which satisfy $F(m) = F(m')$.
3) *Collision resistance*
   It is computationally difficult to find two messages, m and m', with $m \neq m'$, which satisfy $F(m) = F(m')$.

There are three ways of constructing MAC algorithm. First is the MAC construction based on block cipher algorithm (OMAC, CBC-MAC and PMAC). Second is MAC construction based on a cryptographic hash function (HMAC). Third is MAC construction based on hash in general (universal hashing) [5].

B. Cipher Block Chaining Message Authentication Code (CBC-MAC)

CBC-MAC is a method that uses CBC operation mode to construct a MAC based on a block cipher algorithm. CBC-MAC is used to perform compression to messages M that have fixed length mn with a key K, where n is the length of a message block and m is the number of message blocks [4]. Figure 1 shows the CBC-MAC algorithm scheme.

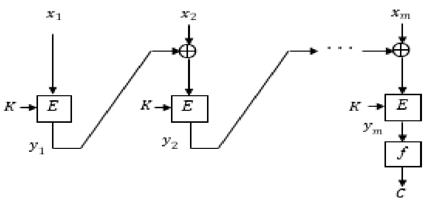

Fig. 1. CBC-MAC algorithm scheme.

C. Encrypted Message Authentication Code (EMAC)

EMAC is a MAC construction which encrypts the result of CBC-MAC with the second key K_2 [4]. EMAC uses 2 keys, K_1 and K_2. Figure 2 shows the EMAC algorithm scheme.

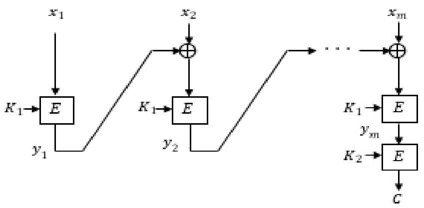

Fig. 2. EMAC algorithm scheme.

D. ECBC-MAC

ECBC-MAC is an improvement of EMAC construction. ECBC-MAC uses 3 keys, i.e. K_1, K_2 and K_3. Figure 3 shows the ECBC-MAC algorithm scheme. For details, we refer to [3].

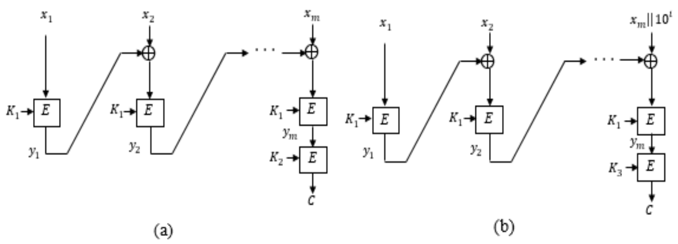

Fig. 3. ECBC-MAC algorithm scheme.

Figure 3 (a) ECBC-MAC is for message that has a multiples length of n and Figure 3 (b) ECBC-MAC for others length.

E. FCBC-MAC

FCBC-MAC is an improvement of ECBC-MAC in terms of efficiency. FCBC-MAC uses 3 keys, i.e. K_1, K_2 and K_3. Figure 4 shows the FCBC-MAC algorithm scheme. For details, we refer to [3].

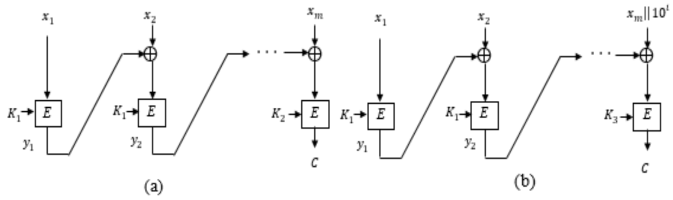

Fig. 4. FCBC-MAC algorithm scheme.

Figure 4 (a) FCBC-MAC is for message that has a multiples length of n and Figure 4 (b) FCBC-MAC for others length.

F. XCBC-MAC

XCBC-MAC is an improvement of ECBC-MAC and FCBC-MAC in term of avoiding the encryption process with many keys because it will cause more subkeys is generated so the computing becomes more severe. XCBC-MAC uses 3 keys i.e. K_1, K_2 and K_3. Figure 5 shows the XCBC-MAC algorithm scheme. For details, we refer to [3].

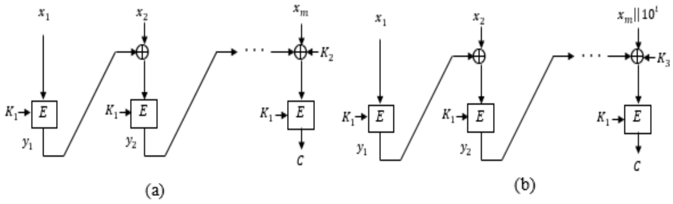

Fig. 5. XCBC-MAC algorithm scheme.

Figure 5 (a) XCBC-MAC is for message that has a multiples length of n and Figure 5 (b) XCBC-MAC for others length.

G. Randomized Message Authentication Code (RMAC)

RMAC is based on CBC-MAC construction that requires a random value generation in the computation. RMAC uses 2 keys, i.e. K_1 and K_2. Figure 6 shows the RMAC algorithm scheme. For details, we refer to [2].

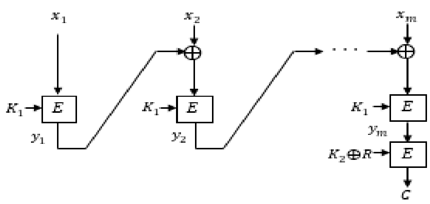

Fig. 6. RMAC algorithm scheme.

H. Two-Key CBC-MAC (TMAC)

TMAC is an improvement of XCBC-MAC. This construction only takes $(k+n)$-bit keys while XCBC-MAC requires $(k+2n)$-bit keys, where k is the length of the key block cipher and n is the length of the block message. TMAC uses 2 keys, i.e. K_1 and K_2. Figure 7 shows the TMAC algorithm scheme. For details, we refer to [6].

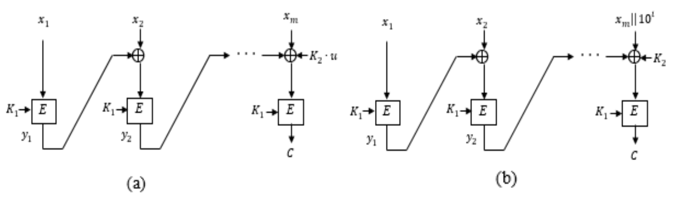

Fig. 7. TMAC algorithm scheme.

Figure 7 (a) TMAC is for message that has a multiples length of n and Figure 7 (b) TMAC for others length.

I. One-Key CBC-MAC (OMAC)

OMAC is a construction that only requires a K key (k-bit). It is the minimum key length that the construction must-have because the underlying block cipher requires a key with k-bit length. OMAC is an improvement of previous construction, TMAC requires $(k+n)$-bit keys and XCBC-MAC requires $(k+2n)$-bit keys, where k is the length of the key block cipher and n is the length of the block message. Figure 8 shows the OMAC algorithm scheme. For details, we refer to [12].

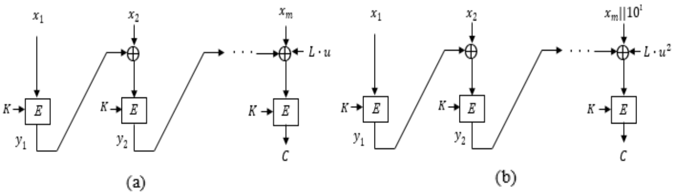

Fig. 8. OMAC algorithm scheme.

Figure 8 (a) OMAC1 is for message that has a multiples length of n and Figure 8 (b) OMAC1 for others length.

J. Cipher-Based Message Authentication Code (CMAC)

CMAC is a construction that relies on a block cipher symmetric key underlying construction. The key used in the CMAC is the key of block cipher itself. That key is used to derive two additional secret value called subkeys i.e K_1 and K_2. The length of both subkeys equal to the length of the block message. Figure 9 shows the CMAC algorithm scheme. For details, we refer to [9].

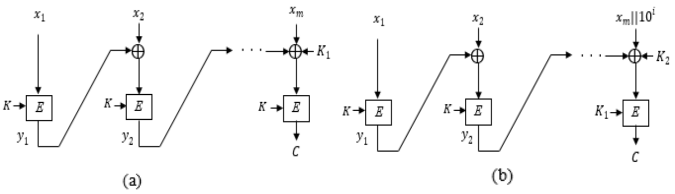

Fig. 9. CMAC algorithm scheme.

Figure 9 (a) CMAC is for message that has a multiples length of n and Figure 9 (b) CMAC for others length.

K. Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) algorithm is a symmetric block cipher that processes data blocks of 128 bits using varied secret key length, i.e. 128 bit, 192 bits, and 256 bits. It processes 128 bits input and yield 128 bits output. A 128 bits sequence is a set of data block. See [8] for more detail discussion.

L. Existential Forgery on CBC-MAC

Existential Forgery on a CBC-MAC is referred to [1]. It consists of 2 stages which can be explained in Figure 10.

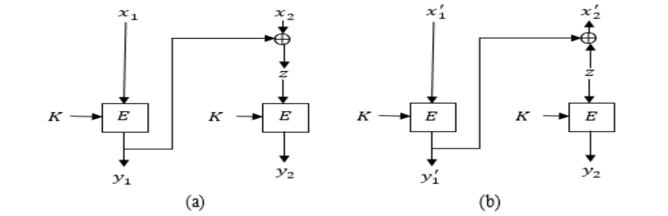

Fig. 10. The first stage (a) and the second stage (b) of existential forgery on a CBC-MAC.

Figure 10 (a) shows the first stage of existential forgery on a CBC-MAC. We can see that the input of the CBC-MAC is x that consists of x_1 and x_2. E is a block cipher encryption function and K is the key of the block cipher. $y_1$and $y_2$are encryption results from the input that enter E. Then calculate z, that satisfy $z = x_2 \oplus y_1$. Afterwards, calculate MAC value of x yield y_2. After that, proceed to the second stage. In the second stage, the input is x' that consists of x_1' and x_2'. In Figure 10 (b), there is the value z which is obtained at the first stage. The value of x_1' is chosen freely, but satisfy $x_1' \neq x_1$. The value of x_2' satisfy $x_2' = z \oplus y_1'$. Then calculate the MAC value of x' yield y_2, which equals to MAC value of x. With the same values z and K, then input either x or x', will generate a value $E_K(z) = y_2$. This can be proven by:

$$x_2' \oplus y_1' = z \oplus y_1' \oplus y_1' = z \quad (1)$$

M. Second Preimage Attack

Second preimage attack is an attack on hash function where the attacker has obtained a message M and its MAC value, C, then attacker formed a different message $M' \neq M$ that satisfy $MAC_K(M') = C$.

III. Proposed Method

The proposed second preimage attack method on various MAC constructions will utilize the concept of existential forgery on CBC-MAC. Assumed that the attacker has an oracle $C = MAC(\cdot)$ that can be used to query any message and has obtained a message $M = x_1||x_2||x_3|| \dots ||x_m$ and its MAC value is C. To get another message M' which has MAC value same to MAC value of message M, an attacker can perform the following steps:

1) Calculate the encryption result of the first block of M yield $y_1 = E_K(x_1)$ (2)

2) Calculate z that satisfy $z = x_2 \oplus y_1$ (3)

3) Formed message $M' = x_1'||x_2||x_3|| \dots ||x_m$ where $x_1' \neq x_1$

4) Calculate the encryption result of the first block of M'yield $y_1' = E_K(x_1')$ (4)

5) Calculate x_2^j that satisfy $x_2' = z \oplus y_1'$ (5)

6) Replace the second block message x_2 of M' with x_2' so that we can obtain $M' = x_1'||x_2'||x_3|| \dots ||x_m$

7) Calculate the MAC value of message M'

The MAC value of M' will same to the MAC value of M. So, the message $M' = x_1'||x_2'||x_3|| \dots ||x_m$ is the second preimage of M.

IV. Results and Analysis

This experiment performed by implementing AES-128 block cipher algorithm into nine various MAC constructions, namely: CBC-MAC, EMAC, ECBC-MAC, FCBC-MAC, XCBC-MAC, RMAC, TMAC, OMAC and CMAC. After that, we apply the proposed second preimage attack method to all that constructions. We use 5 extreme inputs and 5 pseudorandom inputs as the samples which has 640 bits length of each sample. In addition, 640 bits input can be operated without padding process. This experiment uses C programming language and MinGW Dev C++ Compiler.

For each sample, we do 2^{20} minor modifications to first block of sample by increment. Minor modification performed by changing 20 least significant bits of the first block as many as 1048576 possibilities. Every modification to first block followed by the modification to second block appropriated to the proposed method.

After the application of the proposed method for all various constructions, we can obtain the second preimage results to all inputs, both 5 extreme inputs and 5 pseudorandom inputs. Table 1 shows second preimage results of 1 extreme input and 1 pseudorandom input for each construction of all various MAC constructions. The table shows the original message and the second preimage results for each sample. Block 1st and Block 2nd column shows the first block and second block of input respectively. MAC Value column shows the MAC value of the input. All values in Table 1 are represented in hexadecimal. MAC values of each construction shows the second preimages of the original message.

TABLE I. SECOND PREIMAGE RESULT OF EXTREME INPUT AND PSEUDORANDOM INPUT ON CBC-MAC, EMAC, ECBC-MAC, FCBC-MAC, XCBC-MAC, RMAC, TMAC, OMAC AND CMAC CONSTRUCTION

No.	Message			MAC Value		
		Block 1st	***Block 2nd***	**CBC-MAC**	**EMAC**	**ECBC-MAC**
1	Original Message (Extreme)	000000000 000000000 000000000 00000	000000000 000000000 000000000 00000	28a32a13e6 93d5bea71 55f9eebb09 050	783cf0d173 659485b73 a9d79c273 d882	783cf0d173 659485b73 a9d79c273 d882
	Result of Second Preimage	000000000 000000000 000000000 00001	2ae5164c2e 09270c90b 696fe7e693 ba9	28a32a13e6 93d5bea71 55f9eebb09 050	783cf0d173 659485b73 a9d79c273 d882	783cf0d173 659485b73 a9d79c273 d882
		000000000 000000000 000000000 00002	eac845ff52 c17bb1415 5c47786e9 4be6	28a32a13e6 93d5bea71 55f9eebb09 050	783cf0d173 659485b73 a9d79c273 d882	783cf0d173 659485b73 a9d79c273 d882
		⋮	⋮	⋮	⋮	⋮
		000000000 000000000 000000000 fffff	10ae63930c 22662278d 342e62ae1b fb4	28a32a13e6 93d5bea71 55f9eebb09 050	783cf0d173 659485b73 a9d79c273 d882	783cf0d173 659485b73 a9d79c273 d882
		000000000 000000000 000000001 00000	374199635 cdeceed674 fe7b9b026e 254	28a32a13e6 93d5bea71 55f9eebb09 050	783cf0d173 659485b73 a9d79c273 d882	783cf0d173 659485b73 a9d79c273 d882
2	Original Message (Pseudorandom)	c153c96f9 d87c6701c 5f3153ac4 529e4	fe0ad1a2c8 8ec11454f9 b6010fc5ae 96	9a292a399 8beaad3c89 8f2b5e05a7 8f9	d1019f69c0 2392fe8232 11006c90fe a3	d1019f69c0 2392fe8232 11006c90fe a3
	Result of Second Preimage	c153c96f9 d87c6701c 5f3153ac4 529e5	eb6789c5f1 5fa8e8bfb9 64c6ea1c01 0a	9a292a399 8beaad3c89 8f2b5e05a7 8f9	d1019f69c0 2392fe8232 11006c90fe a3	d1019f69c0 2392fe8232 11006c90fe a3
		c153c96f9 d87c6701c 5f3153ac4 529e6	371e7b04cc bb8321ef30 566743d36 eae	9a292a399 8beaad3c89 8f2b5e05a7 8f9	d1019f69c0 2392fe8232 11006c90fe a3	d1019f69c0 2392fe8232 11006c90fe a3
		⋮	⋮	⋮	⋮	⋮
		c153c96f9 d87c6701c 5f3153ac5 529e3	df31dd68d6 7b0a2e0e36 30159211a 475	9a292a399 8beaad3c89 8f2b5e05a7 8f9	d1019f69c0 2392fe8232 11006c90fe a3	d1019f69c0 2392fe8232 11006c90fe a3
		c153c96f9 d87c6701c 5f3153ac5 529e4	786f81d2cd 28a2de7cdb b3504430d 819	9a292a399 8beaad3c89 8f2b5e05a7 8f9	d1019f69c0 2392fe8232 11006c90fe a3	d1019f69c0 2392fe8232 11006c90fe a3

No.	Message			MAC Value		
		Block 1st	***Block 2nd***	**FCBC-MAC**	**XCBC-MAC**	**RMAC**
1	Original Message (Extreme)	000000000 000000000 000000000 00000	000000000 000000000 000000000 00000	28cec3ac99 5b4d18ffb9 5e6c394a0e 51	daf3286c01 85d9b1022 158ec7de44 569	663c20ce48 84e5c8b23 54d174216 0ad5
	Result of Second Preimage	000000000 000000000 000000000 00001	2ae5164c2e 09270c90b 696fe7e693 ba9	28cec3ac99 5b4d18ffb9 5e6c394a0e 51	daf3286c01 85d9b1022 158ec7de44 569	663c20ce48 84e5c8b23 54d174216 0ad5
		000000000 000000000 000000000 00002	eac845ff52 c17bb1415 5c47786e9 4be6	28cec3ac99 5b4d18ffb9 5e6c394a0e 51	daf3286c01 85d9b1022 158ec7de44 569	663c20ce48 84e5c8b23 54d174216 0ad5
		⋮	⋮	⋮	⋮	⋮
		000000000 000000000 000000000 fffff	10ae63930c 22662278d 342e62ae1b fb4	28cec3ac99 5b4d18ffb9 5e6c394a0e 51	daf3286c01 85d9b1022 158ec7de44 569	663c20ce48 84e5c8b23 54d174216 0ad5
		000000000 000000000 000000001 00000	374199635 cdeceed674 fe7b9b026e 254	28cec3ac99 5b4d18ffb9 5e6c394a0e 51	daf3286c01 85d9b1022 158ec7de44 569	663c20ce48 84e5c8b23 54d174216 0ad5
2	Original Message (Pseudorandom)	c153c96f9 d87c6701c 5f3153ac4 529e4	fe0ad1a2c8 8ec11454f9 b6010fc5ae 96	f23168e7fd 2672e3300 a6d5a68cff 7c4	f7858151e8 97b9a3cba8 dc2411f915 0f	ab38e2d88c 732f4d9eea 768ef58661 2f
	Result of Second Preimage	c153c96f9 d87c6701c 5f3153ac4 529e5	eb6789c5f1 5fa8e8bfb9 64c6ea1c01 0a	f23168e7fd 2672e3300 a6d5a68cff 7c4	f7858151e8 97b9a3cba8 dc2411f915 0f	ab38e2d88c 732f4d9eea 768ef58661 2f
		c153c96f9 d87c6701c 5f3153ac4 529e6	371e7b04cc bb8321ef30 566743d36 eae	f23168e7fd 2672e3300 a6d5a68cff 7c4	f7858151e8 97b9a3cba8 dc2411f915 0f	ab38e2d88c 732f4d9eea 768ef58661 2f
		⋮	⋮	⋮	⋮	⋮
		c153c96f9 d87c6701c 5f3153ac5 529e3	df31dd68d6 7b0a2e0e36 30159211a 475	f23168e7fd 2672e3300 a6d5a68cff 7c4	f7858151e8 97b9a3cba8 dc2411f915 0f	ab38e2d88c 732f4d9eea 768ef58661 2f
		c153c96f9 d87c6701c 5f3153ac5 529e4	786f81d2cd 28a2de7cdb b3504430d 819	f23168e7fd 2672e3300 a6d5a68cff 7c4	f7858151e8 97b9a3cba8 dc2411f915 0f	ab38e2d88c 732f4d9eea 768ef58661 2f

No.	Message			MAC Value		
		Block 1st	*Block 2nd*	TMAC	OMAC	CMAC
1	Original Message (Extreme)	000000000 000000000 000000000 00000	000000000 000000000 000000000 00000	a74728f66c 43191fdd65 e6f7f3a06e 78	c5b8395f6d 22c3ea7427 efa4bf3e4d 41	c5b8395f6d 22c3ea7427 efa4bf3e4d 41
	Result of Second Preimage	000000000 000000000 000000000 00001	2ae5164c2e 09270c90b 696fe7e693 ba9	a74728f66c 43191fdd65 e6f7f3a06e 78	c5b8395f6d 22c3ea7427 efa4bf3e4d 41	c5b8395f6d 22c3ea7427 efa4bf3e4d 41
		000000000 000000000 000000000 00002	eac845ff52 c17bb1415 5c47786e9 4be6	a74728f66c 43191fdd65 e6f7f3a06e 78	c5b8395f6d 22c3ea7427 efa4bf3e4d 41	c5b8395f6d 22c3ea7427 efa4bf3e4d 41
		⋮	⋮	⋮	⋮	⋮
		000000000 000000000 000000000 fffff	10ae63930c 22662278d 342e62ae1b fb4	a74728f66c 43191fdd65 e6f7f3a06e 78	c5b8395f6d 22c3ea7427 efa4bf3e4d 41	c5b8395f6d 22c3ea7427 efa4bf3e4d 41
		000000000 000000000 000000001 00000	374199635 cdeceed674 fe7b9b026e 254	a74728f66c 43191fdd65 e6f7f3a06e 78	c5b8395f6d 22c3ea7427 efa4bf3e4d 41	c5b8395f6d 22c3ea7427 efa4bf3e4d 41
2	Original Message (Pseudorandom)	c153c96f9 d87c6701c 5f3153ac4 529e4	fe0ad1a2c8 8ec11454f9 b6010fc5ae 96	0dd49711e 822adce9ae 271caadaad ad6	a23511929 46a4952df8 ebf8598b76 e37	a23511929 46a4952df8 ebf8598b76 e37
	Result of Second Preimage	c153c96f9 d87c6701c 5f3153ac4 529e5	eb6789c5f1 5fa8e8bfb9 64c6ea1c01 0a	0dd49711e 822adce9ae 271caadaad ad6	a23511929 46a4952df8 ebf8598b76 e37	a23511929 46a4952df8 ebf8598b76 e37
		c153c96f9 d87c6701c 5f3153ac4 529e6	371e7b04cc bb8321ef30 566743d36 eae	0dd49711e 822adce9ae 271caadaad ad6	a23511929 46a4952df8 ebf8598b76 e37	a23511929 46a4952df8 ebf8598b76 e37
		⋮	⋮	⋮	⋮	⋮
		c153c96f9 d87c6701c 5f3153ac5 529e3	df31dd68d6 7b0a2e0e36 30159211a 475	0dd49711e 822adce9ae 271caadaad ad6	a23511929 46a4952df8 ebf8598b76 e37	a23511929 46a4952df8 ebf8598b76 e37
		c153c96f9 d87c6701c 5f3153ac5 529e4	786f81d2cd 28a2de7cdb b3504430d 819	0dd49711e 822adce9ae 271caadaad ad6	a23511929 46a4952df8 ebf8598b76 e37	a23511929 46a4952df8 ebf8598b76 e37

From Table 1 we can see that the second preimages of the original message can be obtained in every modification to first block followed by the modification to second block appropriated to the proposed method. As many as 2^{20} modifications for each sample, we can obtain second preimages as many as 2^{20} too.

In Table 1, we can see that the first and the second block is the same to all various MAC constructions. The difference is only at MAC values part. This can be occurred because in the MAC calculation process, the difference only occurs in the final stage. Because the proposed second preimage attack method only exploits the first and the second block while the input length that is used is 640 bits or 5 blocks, so for all constructions, we apply the same treatment to the first and the second block. Different process only occurs on the last message block (5th block), thus producing different MAC values for each construction.

However, there are MAC value similarity of EMAC and ECBC-MAC due to the multiples input length of n, so the calculation process of ECBC-MAC is the same with EMAC (see Fig 2 and Fig 3 (a)). Therefore, we get the same results of second preimage on extreme or pseudorandom inputs obtained at ECBC-MAC and EMAC.

Other similarity also happened on OMAC and CMAC. There are MAC value similarity occured in OMAC and CMAC construction due to the multiples input length of n, so we use OMAC 1 construction which is equivalence with CMAC construction. Thus, we get the same results of second preimage on extreme or pseudorandom inputs obtained at OMAC and CMAC.

The second preimage occurred in our experiments can be proven mathematically as described in Fig 11.

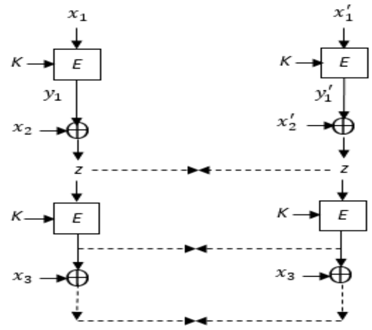

Fig. 11. Collision that occurs in the proposed second preimage attack method.

Fig 11 shows two different messages. The first message in the left, i.e. $x_1||x_2||x_3||$ ... and the second message in the right, i.e. $x_1'||x_2'||x_3||$ ... with the same message block after the second block on both of them. We can see that collision occurrs in point z (the collision is indicated by dotted horizontal lines). Based on the equation (3) i.e.

$$y_1 \oplus x_2 = z \tag{6}$$

and the equation (5), we can obtain

$$y_1' \oplus x_2' = y_1' \oplus (z \oplus y_1') = z \tag{7}.$$

So it can be proven that the collision occurs at the point z i.e.

$$y_1 \oplus x_2 = y_1' \oplus x_2' = z \tag{8}.$$

If the collision occurs in point z while the block messages after the second block are the same, so until the end of the process, both messages will collide. If message $x_1||x_2||x_3||$ ... is considered as the original message, then message $x_1'||x_2'||x_3||$ ... is the second preimage of the original message.

Compare to Jia *et al.* [5], our method is more efficient and effective in finding the second preimage on various MAC constructions. It can be shown as follows. In the second preimage attack method proposed by Jia, to get a second preimage, its require generating two structures that has $2^{(n+1)/2}$ complexity for each structure. So, overall complexity is $2^{(n+3)/2}$ (assumed that g function is ignored). It can be shown in Algorithm 1 [5].

Algorithm 1 Find another pair (x_1', x_2') to make $g(x_1', x_2') = g(x_1, x_2)$

INPUT : x_1, x_2, g.
OUTPUT : x_1', x_2'.
1. $S_1 \leftarrow \emptyset$

Algorithm 1 Find another pair (x_1', x_2') to make $g(x_1', x_2') = g(x_1, x_2)$

2. For $i \leftarrow 0$ to $2^{(n+1)/2}$ do
   Choose $x_1' \notin \{x_1, x_1^0, \dots, x_1^{i-1}\}$ at random and compute $y_1^i \leftarrow g(x_1^i, x_2)$
   $S_1 \leftarrow S_1 \cup \{(x_1^i, y_1^i)\}$
3. For $i \leftarrow 0$ to $2^{(n+1)/2}$ do
   Choose $x_2' \notin \{x_2, x_2^0, \dots, x_2^{i-1}\}$ at random and compute $y_2^i \leftarrow g(x_1, x_2^i)$
   If $y_2^i = y_1^k$ where y_1^k is the second component of an element of S_1, return (x_1^k, x_2^i)

With that complexity, the probability of finding second preimage is 0.63 [5]. Meanwhile, the complexity calculation of our method can be seen in Table 2 (assumed that $E_K(x)$ function is ignored).

TABLE II. THE COMPLEXITY CALCULATION OF SECOND PREIMAGE ATTACK OF OUR METHOD

Process	Complexity
1. Calculate the encryption result of the first block of M yield $y_1 = E_K(x_1)$	1
2. Calculate z that satisfy $z = x_2 \oplus y_1$	1
3. Formed message $M' = x_1'\|\|x_2\|\|x_3\|\| \dots \|\|x_m$ where $x_1' \neq x_1$	1
4. Calculate the encryption result of the first block of M' yield $y_1' = E_K(x_1')$	1
5. Calculate x_2' that satisfy $x_2' = z \oplus y_1'$	1
6. Replace the second block message x_2 of M' with x_2' so that we can obtain $M' = x_1'\|\|x_2'\|\|x_3\|\| \dots \|\|x_m$	1
Complexity Total ***T(n)***	6

Table 2 shows that the complexity of our method to find a second preimage is $T(n) = 6$. Then, the Big-O value is $O(1)$. So, it has a constant complexity which means that our method does not require great complexity as well as on the methods proposed by Jia. Thus, the probability of finding second preimage in our method is 1.

Jia's method has 2 iterations in its process. It is not efficient compare to our method that only has 1.5 iterations. It can be seen in Fig 12.

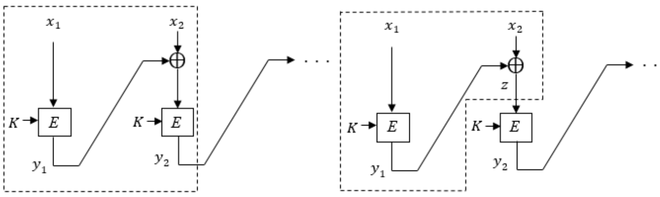

Fig. 12. Iteration process in the method proposed by Jia (left) and the method proposed in this research (right).

In Fig 12, the iteration process is indicated by dotted lines. We can see that the iteration process in Jia's method has done until the second iteration. This is because the method generates 2 message structures, i.e. S_1 which is a set of y_1 and S_2 which is a set of y_2. While the iteration process in our method done until the 1,5 iterations. That is because the method proposed in this study is quite carried out the process up to the point z.

The proposed second preimage attack method does the modifications to the first and the second block, while for the next block is not modified. Therefore, the requirement to apply this method is the minimum message length as many as 2 message blocks.

CONCLUSION

In this paper, we propose the second preimage attack method that utilizes the concept of existential forgery on CBC-MAC on various MAC constructions, i.e. CBC-MAC, EMAC, ECBC-MAC, FCBC-MAC, XCBC-MAC, RMAC, TMAC, OMAC and CMAC. After the application of the method to all of that various MAC constructions with AES-128, the results show that with the modifications as many as 2^{20} for each sample, the second preimages can be obtained easily as many as 2^{20} too. According to the analysis of the results, we conclude that the proposed method in this research is more efficient and effective in finding the second preimage on various MAC constructions compare to Jia's method. Our method will effectively apply using the minimum message length is 2 message blocks.

ACKNOWLEDGMENT.

This research was supported by Sekolah Tinggi Sandi Negara, Bogor, West Java. We would like to thank to the reviewers for their valuable comments.

REFERENCES

[1] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, Handbook of Applied Cryptography, Boca Raton : CRC Press LLC, 1997.

[2] E. Jaulmes, A. Joux and F. Valette, "RMAC A Randomized MAC Beyond The Birthday Paradox Limit," *FSE 2002, LNCS 2365*, pp. 237-251, 2002.

[3] J. Black and P. Rogaway, "CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions," *CRYPTO 2000, LNCS 1880*, pp. 197-215, 2000.

[4] J. Daemen and V. Rijmen, "A New MAC Construction Alred and A Specific Instance Alpha-MAC," *FSE 2005, LNCS 3557*, pp. 1-17, 2005.

[5] K. Jia , X. Wang, Z. Yuan and G. Xu, "Distinguishing Attack and Second-Preimage Attack on the CBC-like MACs," 8th *International Conference*, CANS, 2009.

[6] K. Kurosawa and T. Iwata, "TMAC: Two-Key CBC MAC," *CT–RSA 2003, LNCS 2612*, pp 33-49, 2003.

[7] M. Rjasko, Properties of Cryptographic Hash Functions, Department of Computer Science, Bratislava, 2008.

[8] NIST, Federal Information Processing Standards Publication (FIPS) 197, Springfield : National Institute of Standards and Technology (NIST), 2001.

[9] NIST Special Publication 800-38B, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, 2005.

[10] S. Bakhtiari, R. Safavi-Naini and J. Pieprzyk, Cryptographic Hash Functions: A Survey, Technical Report 95-09, Department of Computer Science, University of Wollongong, 1995.

[11] T. Bartkewitz, Building Hash Function from Block Cipher, Their Security and Implementation Properties, Ruhr-University Bochum, Germany, 2009.

[12] T. Iwata and K. Kurosawa, "OMAC: One-Key CBC MAC," *FSE 2003, LNCS 2887*, pp. 129-153, 2003.