Multicollision attack on CBC-MAC, EMAC, and XCBC-MAC of AES-128 algorithm

To cite this article: Alfonso Brolin Sihite and Bety Hayat Susanti 2017 J. Phys.: Conf. Ser. 893 012048

View the article online for updates and enhancements.

You may also like

- An Electrochemical Sensor Based on Green -AlOOH-Carbonated Bacterial Cellulose Hybrids for Simultaneous Determination Trace Levels of Cd(II) and Pb(II) in Drinking Water. Danfeng Qin, Xun Hu, Yemin Dong et al.

- Compliance boundary conditions for patient-specific deformation simulation using the finite element method. Ece Ozkan and Orcun Goksel

- Mn$_2$O$_3$ Incorporated Carbonized Bacterial Cellulose As Sulfur Host for Lithium-Sulfur Batteries. Vikram Kishore Bharti, Anil Daliprasad Pathak, Mudrika Khandelwal et al.
Multicollision attack on CBC-MAC, EMAC, and XCBC-MAC of AES-128 algorithm

Alfonso Brolin Sihite and Bety Hayat Susanti
Sekolah Tinggi Sandi Negara
E-mail: alfonso.brolin.s.17@gmail.com, bety.hayat@stsn-nci.ac.id

Abstract. A Message Authentication Codes (MAC) can be constructed based on a block cipher algorithm. CBC-MAC, EMAC, and XCBC-MAC constructions are some of MAC schemes that used in the hash function. In this paper, we do multicollision attack on CBC-MAC, EMAC, and XCBC-MAC construction which uses AES-128 block cipher algorithm as basic construction. The method of multicollision attack utilizes the concept of existential forgery on CBC-MAC. The results show that the multicollision can be obtained easily in CBC-MAC, EMAC, and XCBC-MAC construction.

1. Introduction
Hash function is the function that process and maps an arbitrary length string to a fixed output length as a hash value [1]. During a transmission, the error of the message can be detected by the hash function by attaching the hash value with the message. If the hash value of the received message was not the same as the hash value that attached, then there was an error in the message [2].

Hash functions are categorized into unkeyed hash function (Modification Detection Codes) and keyed hash function (Message Authentication Codes). In addition to integrity, Message Authentication Codes (MAC) also guarantees authentication of data origin because it uses keys in the process [3]. MAC can be designed by defining a new mode of operation for existing primitive [4]. Some of them are the Cipher Block Chaining Message Authentication Code (CBC-MAC), Encrypted Message Authentication Code (EMAC), and XCBC-MAC. In this research, we use CBC-MAC, EMAC, and XCBC-MAC construction for constructing the MAC.

In 2001, National Institute of Standards and Technology (NIST) specified Advanced Encryption Standard (AES) as a block cipher standard algorithm [5]. AES has been proved to have good security and easy to implement as a block cipher algorithm [4]. AES process 128 bit data block using varied secret key length, i.e. 128-bit, 192-bit, and 256-bit. AES with 128-bit key called AES-128 [5]. In this research, we implement AES-128 in CBC-MAC, EMAC, and XCBC-MAC construction.

A hash function must satisfy three main requirements, i.e. preimage resistance, second preimage resistance, and collision resistance to be a good hash function [6]. Collision resistance is a condition where it is computationally difficult to find two messages $x$ and $x'$ with $x \neq x'$ which satisfy $h(x) = h(x')$ [3]. In this research, a multicollision attack will be applied on CBC-MAC, EMAC, and XCBC-MAC of AES-128 algorithm. The method of multicollision attack will utilize the concept of existential forgery on CBC-MAC. The goal of this research is to find the multicollision on CBC-MAC, EMAC, and XCBC-MAC construction based on a block cipher algorithm.
2. Theoretical Background

2.1. Hash Function
A hash function takes an arbitrary length string as an input then maps it to a fixed output length called as a hash value [1]. A hash value can be used as an image representation of an input string if it can be uniquely identified with an input string. That is the basic idea of a cryptographic hash function [3].

A hash function must satisfy the following three properties, i.e: preimage resistance (given a hash value \( y \), then it is computationally difficult to find \( m \) which satisfy \( F(m) = y \)), second preimage resistance (given \((m, F(m))\), it is computationally difficult to find \( m \), with \( m \neq m' \), which satisfy \( F(m) = F(m') \)), and collision resistance (it is computationally difficult to find two messages \( m \) and \( m' \), which satisfy \( F(m) = F(m') \)) [3]. There are three ways to construct a MAC algorithm, i.e. based on a block cipher algorithm (OMAC, CBC-MAC and PMAC), based on a cryptographic hash function (HMAC), and based on hash in general (universal hashing)[7].

2.2. Cipher Block Chaining Message Authentication Code (CBC-MAC)
CBC-MAC is a Message Authentication Code (MAC) based on block cipher with Cipher Block Chaining (CBC) mode. CBC-MAC performs compression to messages \( M \) (fixed length \( mn \)) with a key \( K \), where \( n \) is the length of a message block and \( m \) is the number of message blocks [7]. For details, we refer to [7].

2.3. Encrypted Message Authentication Code (EMAC)
EMAC is a Message Authentication Code (MAC) construction which encrypt the result of CBC-MAC with the second key \( K_2 \) [7]. EMAC performs compression to messages \( M \) that have fixed length with two keys \( K_1 \) and \( K_2 \), where \( n \) is the length of a block message and \( m \) is the number of message blocks. We refer to [7] for details.

2.4. XCBC-MAC
XCBC-MAC is an improvement of MAC construction of CBC-MAC that uses 3 keys i.e. \( K_1 \), \( K_2 \), and \( K_3 \). See [4] for more detail discussion.

2.5. Advanced Encryption Standard (AES)
Advanced Encryption Standard (AES) is a symmetric block cipher algorithm that performs encryption (encipher) and decryption (decipher) of information. It processes 128 bit data block using varied secret key length, i.e. 128 bit, 192 bit, and 256 bit, and has 10, 12, 14 rounds which called as AES-128, AES-192, and AES-256 respectively. Either AES-128, AES-192, and AES-256 processes 128 bits input and yield 128 bit output. The AES algorithm performs key expansion session before encryption or decryption process. For details, we refer to [5].

2.6. Existential Forgery on CBC-MAC [3]
Existential Forgery on a CBC-MAC consists of 2 stages. Fig 1 is the first stage and Fig 2 is the second stage.

Fig 1 shows the first stage of existential forgery on a CBC-MAC. The input of the CBC-MAC is \( x = x_1 || x_2 \). \( E \) is a block cipher encryption function and its key is \( K \). The encryption results are \( y_2 \) and \( y_1 \). The stage begin by calculating \( z \) that satisfy \( z = x_2 \oplus y_1 \). Then, calculate MAC value of \( x \) yield \( y_2 \). After that, it is begin to the second stage that is explained in Fig 2.

We can see that the input of the second stage is \( x' = x'_1 || x'_2 \). \( x'_1 \) is randomly chosen but satisfy \( x'_1 \neq x_1 \). \( x'_2 \) is determined by \( x'_2 = z \oplus y'_1 \) where \( z \) is obtained in the first stage. After that, calculate the MAC value of \( x' \) yield \( y_2 \). It is equals to the MAC of \( x \). Both \( x \) and \( x' \) will generate \( E_K(z) = y_2 \) as long as the values of \( z \) and \( K \) are the same. It can be proven by:

\[
x'_2 \oplus y'_1 = z \oplus y'_1 \oplus y'_1 = z
\]
2.7. Multicollision Attack
Multicollision attack is finding different inputs \(\{M_1, M_2, \ldots, M_s\} \subseteq M\) that satisfy \(f(M_1) = f(M_2) = \ldots = f(M_s)\) [8]. In finding collisions, the padding process can be ignored if the messages that are used have the same length even though the messages are different because the padding process to both messages will yield the same pad. Moreover, if the intermediate hash chaining values collide at some point in the hash computation of two different messages and the next message blocks are same, at the end of the hash computation will yield the same hash value (collision). Thus, for the messages that have the same length, collisions that happened on the unpadded messages clearly lead to collisions on the padded messages [9].

3. Research Method
The multicollision attack method to CBC-MAC, EMAC, and XCBC-MAC utilize the concept of existential forgery on CBC-MAC. The multicollision in this research is four different messages that collide (4-collision). Assumed the adversary has the oracle \(C = MAC()\) and has generated a message \(M = x_1||x_2||x_3||\ldots||x_m\). To get four messages that have same MAC, the adversary can apply the multicollision attack method, i.e.:

(i) Calculate the encryption result of the first block of message \(M = x_1||x_2||x_3||x_4\ldots||x_m\) yield \(y_1 = E_K(x_1)\).

(ii) Calculate \(z\) that satisfy \(z = x_2 \oplus y_1\).

(iii) Formed message \(M' = x_1||x_2||x_3||x_4\ldots||x_m\) where \(x_1' \neq x_1\).

(iv) Calculate the encryption result of the first block of message \(M' = x_1'||x_2||x_3||x_4\ldots||x_m\) yield \(y_1' = E_K(x_1')\).

(v) Determine \(x_2'\) that satisfy \(x_2' = z \oplus y_1'\).

(vi) Replace the second block message \(x_2\) of message \(M'\) with \(x_2'\) so that we can obtain message \(M'' = x_1'||x_2'||x_3||x_4\ldots||x_m\).

(vii) Then, on message \(M'\), calculate the chaining process until the third block yield \(y_3 = E_K(E_K(E_K(x_1) \oplus x_2) \oplus x_3)\).

(viii) Calculate \(z'\) that satisfy \(z' = x_4 \oplus y_3\).

(ix) Formed message \(M''' = x_1||x_2||x_3'||x_4\ldots||x_m\) where \(x_3' \neq x_3\).
(x) On message $M''$, calculate the chaining process until the third block yield $y^k_3 = E_K(E_K(x_1 \oplus x_2) \oplus x_3^k)$. 
(xi) Determine $x'_4$ that satisfy $x'_4 = y'_5 \oplus y^k_3$. 
(xii) Replace the fourth block message $x_4$ of message $M''$ with $x'_4$ so that we can obtain message $M' = x_1 || x_2 || x'_3 || x'_4 || x_5 || ... || x_m$. 
(xiii) From the messages $M$, $M'$, and $M''$, four messages can be constructed that will collide (4-collision). The four messages are:

$$M_1 = x_1 || x_2 || x_3 || x_4 || x_5 || ... || x_m;$$
$$M_2 = x_1 || x_2 || x_3 || x'_4 || x_5 || ... || x_m;$$
$$M_3 = x'_1 || x'_2 || x'_3 || x_4 || x_5 || ... || x_m;$$
$$M_4 = x'_1 || x'_2 || x'_3 || x'_4 || x_5 || ... || x_m.$$ 

The four message $M_1$, $M_2$, $M_3$, and $M_4$ will satisfy: $MAC(M_1) = MAC(M_2) = MAC(M_3) = MAC(M_4)$.

This experiment performed by implementing AES-128 block cipher algorithm into CBC-MAC, EMAC, and XCBC-MAC construction using C programming language and MinGW Dev C++ Compiler. After that, we apply the multicollision attack to that constructions. The number of samples that is used are 5 extreme and pseudorandom inputs which has 640 bits length of each sample. In addition, 640 bits can be operated without padding process.

For each sample, we do $2^{20}$ minor modifications to the original input. For the first block of the sample, we do a modification by increment. The second block is determined appropriated to the multicollision attack method. For the third block, minor modification performed by changing 20 least significant bits as many as 1048576 possibilities. Every modification to third block followed by the modification to fourth block appropriated to the multicollision attack method.

4. Results and Analysis

After the application of the multicollision attack method for CBC-MAC, EMAC, and XCBC-MAC constructions, we can obtain the multicollision results to all inputs, both 5 extreme and pseudorandom inputs. Table I shows multicollision results of 1 extreme and 1 pseudorandom input for each construction of CBC-MAC, EMAC, and XCBC-MAC. The table shows the 4 messages which are multicollision (4-collision) that obtained as many as $2^{20}$ for each sample. Change column shows the order of the changes of a sample. Block 1st, Block 2nd, Block 3rd, and Block 4th column shows the first, second, third, and fourth block of input, respectively. MAC Value column shows the MAC value of the input. All values in Table I are represented in hexadecimal. MAC values of each construction show the multicollision (4-collision) that obtained.

From Table 1, we can see that the multicollision (4-collision) can be obtained in every modification to the third block followed by the modification to fourth block appropriated to the multicollision attack method for each sample of each MAC construction. As many as $2^{20}$ modifications for each sample, we can obtain the multicollision (4-collision) as many as $2^{20}$ too. But, we can see that the first and second block is the same for every modification. It is because we just make a modification to the first and the second block.

CBC-MAC, EMAC, and XCBC-MAC constructions have been analyzed to find the reason why the multicollision (4-collision) can be obtained easily. It can be proven mathematically and to facilitate it, we can see Fig 3.

As shown in Fig 3, there are two different messages. The first message in the left i.e. $x_1 || x_2 || x_3 || x_4 || x_5 || ...$ and the second message in the right i.e. $x'_1 || x'_2 || x'_3 || x'_4 || x_5 || ...$ with block messages after the fourth block on these two messages are the same. We can see that collision occurs in point $z$ and $z'$ (the collision is indicated by dotted horizontal lines). The proof of collision that occurred in point $z$ can be explained as follows. Based on the equation (3) and the equation (5), we can obtain $y^l_1 \oplus x_2 = y^l_1 \oplus (z \oplus y^l_1) = z$. 

$$y^l_1 \oplus x_2 = y^l_1 \oplus (z \oplus y^l_1) = z. \quad (10)$$
### Table 1. Multicollision (4-Collision) Result of Extreme Input and Pseudorandom Input on CBC-MAC, EMAC, and XCBC-MAC Construction

| Sample Change | CBC-MAC | EMAC | XCBC-MAC |
|---------------|---------|------|----------|
| 1             | 783cfd17365 | 9485b73a9d79 | db91022158ec |
| 2^{20}        | 783cfd17365 | 9485b73a9d79 | db91022158ec |
| 1             | 783cfd17365 | 9485b73a9d79 | db91022158ec |
| 2^{20}        | 783cfd17365 | 9485b73a9d79 | db91022158ec |
| e153c96f8d7ec | b7c147df054 | 9c292a3998be | d10f19f0e023 |
| 6701c5f315a3 | 6e28a73a3e9 | aed389a28f6e | 92f3e3c821100 |
| 4529c4       | e3a9e6    | 11a367b9   | 05a789    |
| e153c96f8d7ec | b7c147df054 | 9c292a3998be | d10f19f0e023 |
| 6701c5f315a3 | 6e28a73a3e9 | aed389a28f6e | 92f3e3c821100 |
| 4529c4       | e3a9e6    | 11a367b9   | 05a789    |
| e153c96f8d7ec | b7c147df054 | 9c292a3998be | d10f19f0e023 |
| 6701c5f315a3 | 6e28a73a3e9 | aed389a28f6e | 92f3e3c821100 |
| 4529c4       | e3a9e6    | 11a367b9   | 05a789    |

*Note: The table continues with similar entries for various message blocks.*
Figure 3. Collisions in multicollision attack method to CBC-MAC, EMAC, and XCBC-MAC constructions.

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

$$y_1 \oplus x_2 = y_1^i \oplus x_2^j = z. \tag{11}$$

For the proof of collision that occurred in point $z'$ can be explained as follows. Based on the equation (7) and the equation (9), we can obtain

$$y_3^k \oplus x_4^l = y_3^k \oplus (z' \oplus y_3^k) = z'. \tag{12}$$

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

$$y_3 \oplus x_4 = y_3^k \oplus x_4^l = z'. \tag{13}$$

If the collisions occur in point $z$ and $z'$ while the block messages after the fourth block are the same, so until the end of the process, both messages will collide. From these two messages, we can formed four messages that will collide at the end of the process which will become the multicollision (4-collision), i.e.:

$$M_1 = x_1 || x_2 || x_3 || x_4 || \ldots || x_m;$$
$$M_2 = x_1 || x_2 || x_3^k || x_4^l || \ldots || x_m;$$
$$M_3 = x_1^i || x_2^j || x_3 || x_4 || \ldots || x_m;$$
$$M_4 = x_1^i || x_2^j || x_3^k || x_4^l || \ldots || x_m.$$
Thus, the multicollision (4-collision) can be obtained easily in CBC-MAC, EMAC, and XCBC-MAC constructions by modifications to four blocks of the message.

5. Conclusion
In this paper, we do multicollision attack on CBC-MAC, EMAC, and XCBC-MAC which use AES-128 as the block cipher. The results show that by the modifications as many as $2^{20}$ for each sample of each MAC construction, the multicollisions (4-collision) can be obtained as many as $2^{20}$ too. To obtain at least a multicollision (4-collision), it is easily by modifications to four blocks of the message in CBC-MAC, EMAC, and XCBC-MAC constructions.

Acknowledgements
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] Bartkewitz T 2009 Building Hash Function from Block Cipher, Their Security, and Implementation Properties Ruhr-University Bochum Germany
[2] Bakhtiari S, Safavi-Naini R and Pieprzyk J 1995 Cryptographic Hash Function : A Survey Technical Report 95-09 Department of Computer Science University of Wollongong
[3] Menezes A J, Van Oorschot P C and Vanstone S A 1997 Handbook of Applied Cryptography (Boca Raton: CRC Press LLC)
[4] Black J and Rogaway P 2000 CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions 20th Int. Crypto. Conf. CRYPTO 2000 LNCS 1880 p 197-215
[5] NIST 2001 Federal Information Processing Standards Publication (FIPS) 197 Springfield : National Institute of Standards and Technology (NIST)
[6] Rjasko M 2008 Properties of Cryptographic Hash Functions Department of Computer Science Bratislava
[7] Jia K, Wang X, Yuan Z and Xu G 2009 Distinguishing Attack and Second-Preimage Attack on the CBC-like MACs 8th Int. Conf. CANS
[8] Upadhyay J K 2010 Generic Attack on Hash Function Thesis of University of Waterloo
[9] Joux A 2004 Multicollisions In Iterated Hash Function Application To Cascaded Construction 24th Int. Crypto. Conf. CRYPTO 2004 LNCS 3152 p 306-316