Received: July 2021 Received in revised form: September 2021 Accepted: November 2021

Performance Analysis of Block Chaining Message Authentication Code (CBC MAC) and its Variants

- **Chhetra Bahadur Chhetri** [1]

Abstract

The cryptographic algorithms employed in internet security must be able to handle packets which may vary in size over a large range. The majority of cryptography algorithms divide messages into large blocks to process them. due to this fact the messages must be prepared by padding the desired amount of zero bits to induce an integer number of blocks. This process contributes a considerable overhead when the short messages are more dominant within the message stream. In this paper, analyze is targeted on the performance of varied message authentication code generator algorithm supported cipher block. These all variants of cipherbased must share symmetric key before creating message authentication code. All variants of CBC MAC are implemented in JAVA. The results of empirical performance shows that two variants namely TMAC perform better for AES Encryption algorithm in larger size otherwise EMAC show the higher result with the Triple DES symmetric algorithm. The result shows that, when consider only on the performance aspect. Cycle/byte is calculated for comparing different variants of CBC MAC. Cycle/byte is decreased when input size of message is increased. Advanced Encryption Standard (AES) algorithm shows good performance than TDES and its better safety features than DES. CBC-MAC is maybe visiting be standardized as an AES mode of operation.

Keywords: CBC-MAC, EMAC, TMAC, Triple DES, Advanced Encryption Standard, Cycle/byte

1. Introduction

Message authentication has two parts: source authentication, which checks the source's identity and prevents messages from being accepted from a malicious source, and data integrity, which prevents the data from being changed with. One way of achieving authenticity as well as integrity is creation of message authentication code abbreviation as MAC or cryptographic checksum. MAC

[1] Mr. Chheri is the faculty member of Department of Information &Technology of National College of Computer Studies

is a small fixed-size block of data which is computed by using a secret key that share between communicating parties. The Cipher Block Chaining Message Authentication Code (CBC MAC) is a well-known block cipher-based means of generating a message authentication code (MAC). The CBC MAC, on either hand, is well known for being insecure unless the message length is fixed. As a result, several CBC MAC versions for variable length messages have been proposed. The Encrypted MAC (EMAC) was the first to be proposed. It is obtained by encrypting the CBC MAC value with a new key K_2 that use the encryption function. That is:

$$EMAC_{k1,\,k2}(M) = E_{k2}(CBC_{k1}(M))$$

Where M is a message, K_1 is the key of the CBC MAC and CBC_{k1} (M) is the CBC MAC value of (Petrank & Rackoff, 2000) then proved that EMAC is secure if the message length is a positive multiple of n. EMAC requires two key scheduling of the underlying block cipher E. where E represents encryption function. XCBC takes three keys: one block cipher key K_1, and two n-bit keys K_2 and K_3. TMAC is formed from XCBC by changing (K_2, K_3) with (K_2.u, K_2), where u is a non-zero constant and "." in GF(2^n) represents multiplication. Whereas, in GF(2^n), OMAC is produced by substituting (K_2, K_3) with (L.u, L.u2) for some non-zero constant u, where L is supplied by $E_k(0^n)$ (John & Phillip, 2000).

2. Research Methodology

Various sample messages for different size are fed to the different variant of CBC MAC and the enciphered/deciphered messages calculated along with the key required in order to compare the performance of candidate algorithms used within CBC MAC. The algorithms used for various flavors of CBC MAC are implemented in JAVA. The candidates algorithms allowed to run on J2SETM (Java™ Platform, Standard Edition 7 Development Kit) environment with Windows7, 64 bits machine having 4GB RAM, with Intel CORETM i5 processor and analyze the performance of CBC MAC and its Variants using different Symmetric Cryptographic algorithm like Triple DES and AES in order to gain the efficiency over existing method like “Encrypted MAC (EMAC), XCBC MAC, Two-Key CBC MAC (TMAC) and One-Key CBC MAC (OMAC1). In this paper Cycle/Byte calculation is performed.

3. Literature review

In the current scenario, most of the researchers have proposed different message authentication code algorithms to make more secure and fast. Some of them are parallelizable some are using secure hash function like SHA- 1, SHA -2. Several research works to prove the security bounds for each variant of CBC MAC. Some of those literatures have analyzed MAC as pseudorandom function (PRF) similarly Pseudo-Random Functions and Parallelizable Modes of Operations of a Block Cipher have been proposed (Sarkar, 2009). Most of researchers have proposed improved security bounds in parallelizable environment as PMAC and non-parallelizable cipher bock based TMAC and XCBC (Rogaway, 2000)

Many studies have been done on MAC built using block cipher including CBC MAC and its variants. They have been studied the security bounds for each variant of CBC. Performance analysis is done by comparing their key size and number of encryption invocation needed for creating MAC. Number of keys scheduled for each algorithm is compared with their security proposed.

3.1 Message Authentication Code (MAC)

Commonly, Message authentication code (MAC), also referred to as a keyed hash function is used to achieve message authentication. Generally, MACs are used between those two parties who share a secret key to authenticate information exchanged between them. A MAC function uses a secret key and a data block as input which produces a hash value, knows as the MAC. This can then be either transmitted with the protected message or stored with it. If the message's integrity needs to be verified, the MAC function can be applied to it and the result compared to a previously saved MAC value. Without knowing the secret key, an attacker who modifies the message will be unable to change the MAC value. Because no one else has access to the secret key, the verifying party knows who the sending party is as well. In practice, MAC algorithms are meant to be more efficient than encryption techniques. A secret key is used to construct a short fixed-size block of data known as a cryptographic checksum or MAC, which is attached to the message as an alternative authentication approach. This method presupposes that two communication parties, say A and B, have a shared secret key. When A needs to transmit a message to B, it works out the MAC as a function of the message and the key: (Stallings, 2010) MAC = MAC (K, M) Where M = input message C = MAC function K = shared secret key MAC

= message authentication code

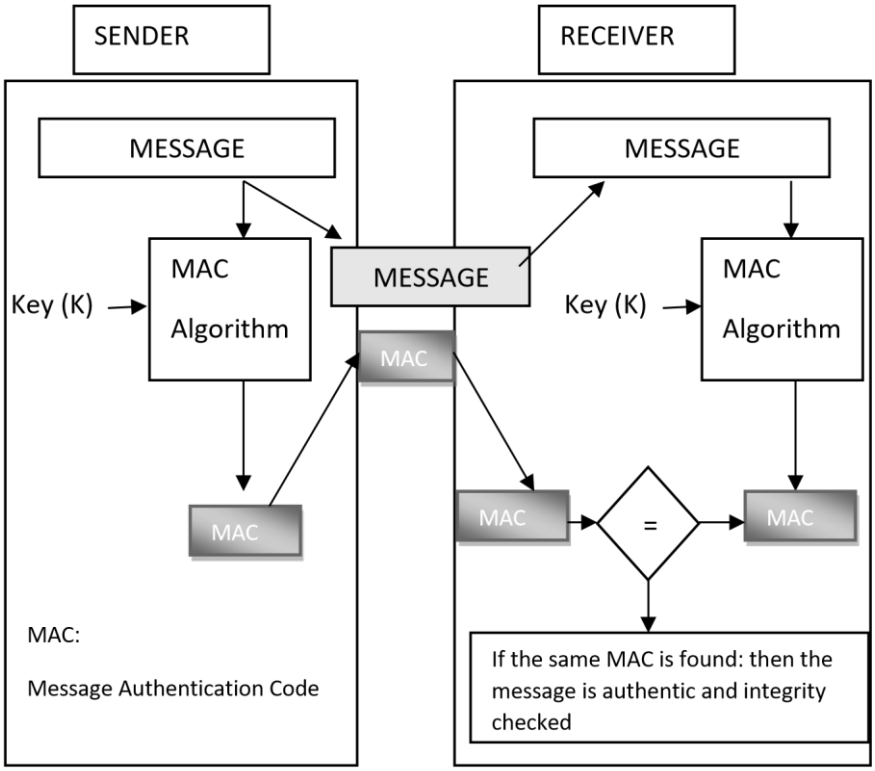

Figure 1: Message Authentication Code

3.2 Key Generation

The key length of CBC MAC is k bit only where k is the bit length of secrete key share by sender and receiver but key length for other variants is vary than k bit. Where EMAC required 2k bit obtained from two secrete key. XCBC required three key one k bit and other two key have 2n bits, here n is the length of block size of message. Suggested by (Kurosawa & Iwata, 2003/082) it is reduced as following:

K1 = the first k bits of $E_k(C_{1a}) \| E_k(C_{1b})$,

K2 = $E(C_2)$,

K3 = $E(C_3)$. Where C_{1a}, C_{1b}, C_2, C_3 are distinct constants.

Next two variants TMAC and OMAC are the refinement of XCBC. Key for TMAC is obtained by replacing $K_{2.u}$ and keys are generated in OMAC by replacing K_2 as L.u and K_3 by $L.u^2$. Where u be the some positive constant and "." is a multiplication in $GF(2^n)$ and L is obtained as follows:

L = $E_k(0^n)$ (Stallings, 2010).

3.3 Cipher Block Chaining (CBC)

CBC is a technique that produces multiple ciphertext blocks when the same plaintext block is repeated. The XOR of the current plaintext block and the preceding ciphertext block is used as the input to the encryption algorithm in this method; the same symmetric key is utilized for each block. In effect, the processing of the plaintext blocks has been linked together. Each plaintext block's input to the encryption process has no defined link to the plaintext block. As a result, repetitive patterns of bits are hidden. If the last block is a partial block, the CBC mode, like the ECB mode, requires that it be padded to full bits. Each cipher block is put through the decryption algorithm for decryption. Each cipher block is put through the decryption algorithm for decryption. The plaintext block is created by XORing the result with the previous ciphertext block. (John & Phillip, 2000) It can be viewed as

$C_j = E(K, [C_{j-1} \oplus P_j])$

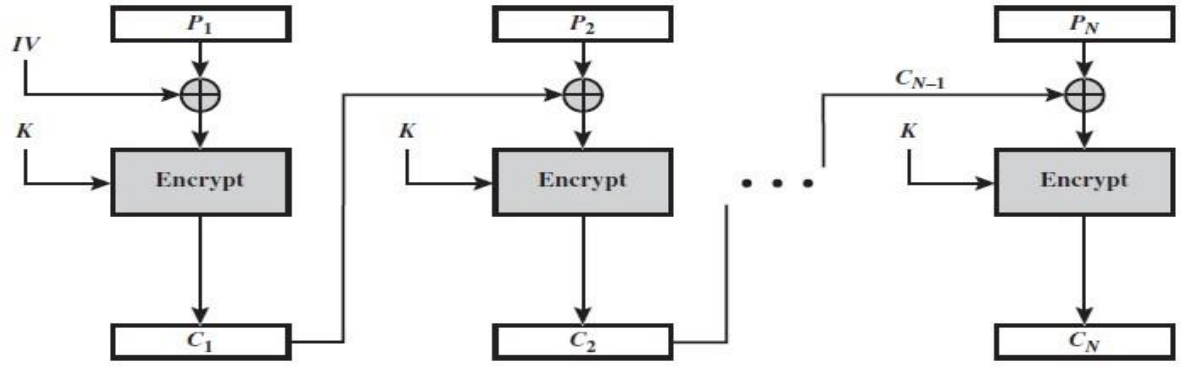

Figure 2 : Encryption of CBC Mode

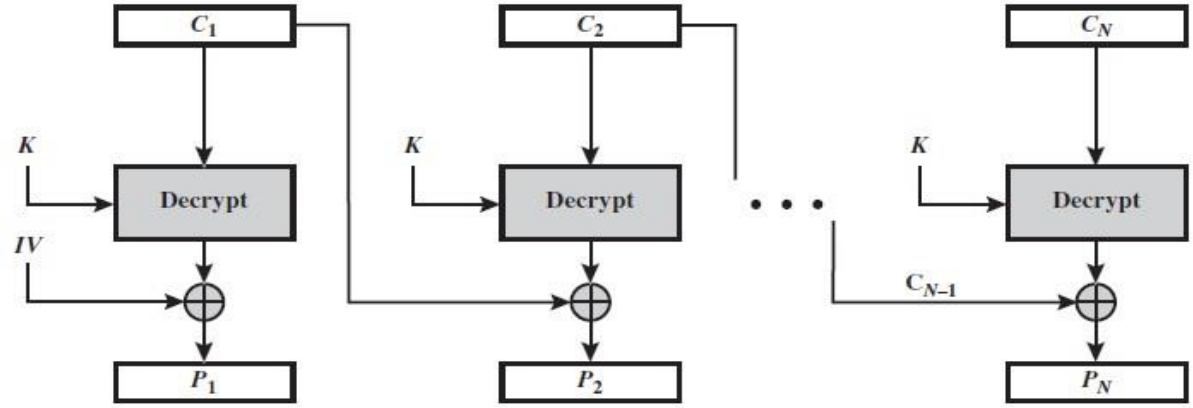

Figure 3 : Decryption of CBC mode

3.3.1 Cipher Block Chaining Message Authentication Code (CBC -MAC)

CBC-MAC is a block cipher-based approach for generating a message authentication code. To form a chain of blocks, the message is divided into an equal number of blocks and encrypted with a block cipher algorithm in CBC mode, with each block depending on the proper encryption of the preceding block. For this interdependence, every change to the plaintext bits causes the final encrypted block to change in unexpected ways that can't be predicted without knowing the block cipher's secret key. M can be decomposed into blocks as $M = M_1, M_2 \ldots M_m$, where M_i denotes a message block. The result is XORed with the following block after passing each block through the encryption E with key K If E_k denotes encryption with a secret key K, then the cipher block chaining technique is as follows: (John & Phillip, 2000)

Algorithm

INPUT: block of message (M)

OUTPUT: MAC (Tag)

Algorithm CBC-MAC_K (M)

Partition M into M [1]……M[m]

C [0] = 0^n for i =

1 to m do

$C[i] = E_K(C[i-1] \oplus M[i])$

Tag = C[m] return Tag

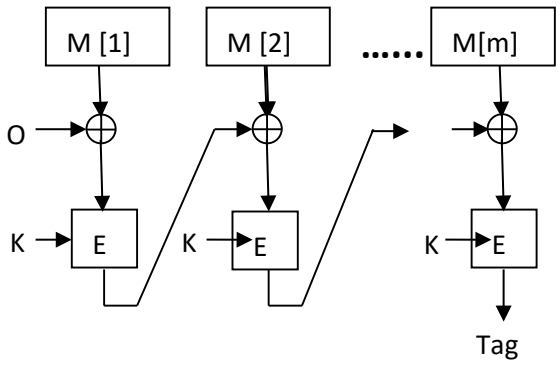

Figure 4 : CBC -MAC

The CBC-MAC is available in several versions, each with its own set of features such as padding and length variable. The most common method of padding for CBC-MAC is to treat the final input block as a partial block of data that is left justified and has zeroes appended to make a full block.

3.3.2 Encrypted –MAC (EMAC)

Encrypted MAC (EMAC) was created to deal with changeable message length in blocks m. CBC E_{K1} (M) is encrypted by MAC using a new block cipher key K2. That is:

$$EMAC_{EK1,EK2}(M) = E_{K2}\,(CBC_{EK1}(M)).$$

The RACE project generated the EMAC, a popular variation of the CBC-MAC. One issue is that the message length must be a positive multiple of n, hence the domain must be $((0, 1)^n)^+$. To deal with messages whose lengths are not multiples of n, append the minimum 10^i to M as padding. Even if the message's size is already a multiple of n, the padding is added.

The domain of EMAC is 0,1*, which means it can handle entirely changeable message lengths.

Algorithm

INPUT: block of message (M)

OUTPUT: MAC (Tag)

Algorithm $EMAC_{K1, K2}$ (M)

Partition M into M [1]……M[m]

C [0] = 0^n for i =

1 to m do

C[i] = E_{K1}(C [i-1] ⊕ M[i])

Tag = E_{K2} (E_{K1}(C [m-1] ⊕ M[m])) return

Tag

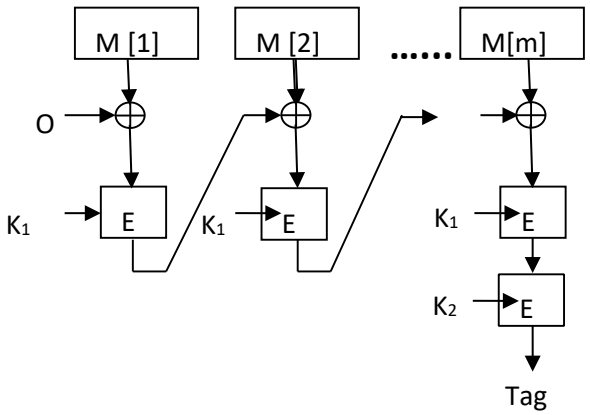

Figure 5: EMAC

3.3.3 XCBC MAC

XCBC takes three keys: one block cipher key K_1, and two n –bit keys K_2 and K_3. XCBC makes two cases to deal with arbitrary length messages: $M \in (\{0, 1\})^+$ and $M \notin (\{0, 1\})^+$. If $M \in (\{0, 1\})^+$ then XCBC computes exactly the same as CBC MAC, except XORing an n-bit key K_2 before encrypting the last block. If $M \notin (\{0, 1\})^+$ then minimal 10^i padding ($i \geq 0$) is appended to M so that the length is a multiple of n, and XCBC computes exactly the same as the CBC MAC, except XORing another n-bit key K_3 before encrypting the last block.

Algorithm

INPUT: block of message (M)

OUTPUT: MAC (Tag)

Algorithm $XCBCMAC_{K1, K2, K3}$ (M)

Partition M into M [1]……M[m]

C [0] = 0^n for i = 0

to m-1 do

C[i] = E_{K1}(C [i-1] ⊕ M[i]) if | M[m] | = n then Tag = E_{K1} (C [m-1] ⊕ M[m] ⊕ K2) else Tag = E_{K1} (C [m-1] ⊕ M[m] 10....0 ⊕ K3) return Tag

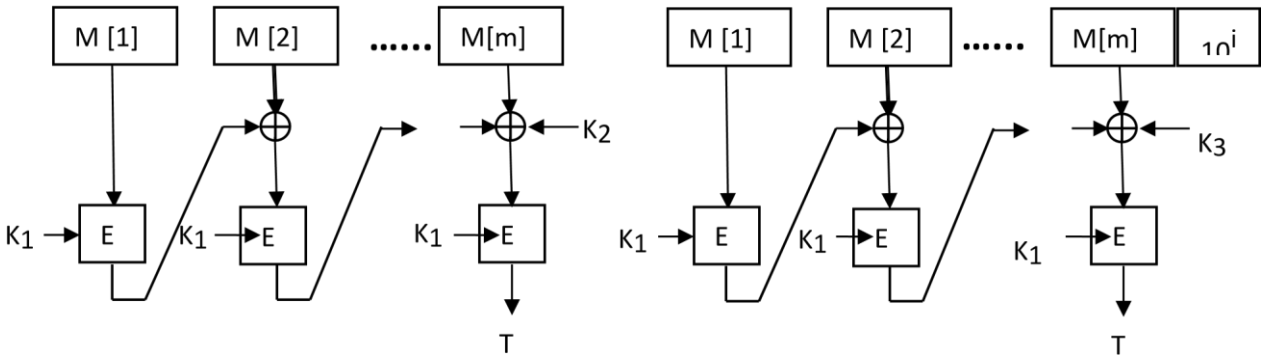

Figure 6: XCBC

3.3.4 TMAC

A refinement of XCBC is the two-key CBC Message authentication code. TMAC requires only a (k+n) bit key, whereas XCBC needs a (k+2n) bit key, where k is the underlying block cipher's key

length and n is the block length. TMAC is obtained by substituting (K_2, K_3) with (K_2.u, K2) in XCBC, where u is a non-zero constant and "." denotes multiplication in GF(2^n) (Iwata & Kurosawa, 2003)

Algorithm

INPUT: block of message (M)

OUTPUT: MAC (Tag)

Algorithm TMAC$K_{1, K2}$ (M)

Partition M into M [1] ……M[m]

C [0] = 0^n

for i = 0 to m-1 do

C[i] = E_{K1}(C [i-1] ⊕ M[i]) if | M[m] | = n then Tag = E_{K1} (C [m-1] ⊕ M[m] ⊕ K2.u) else Tag = E_{K1} (C [m-1] ⊕ M[m] 10....0 ⊕ K2) return Tag

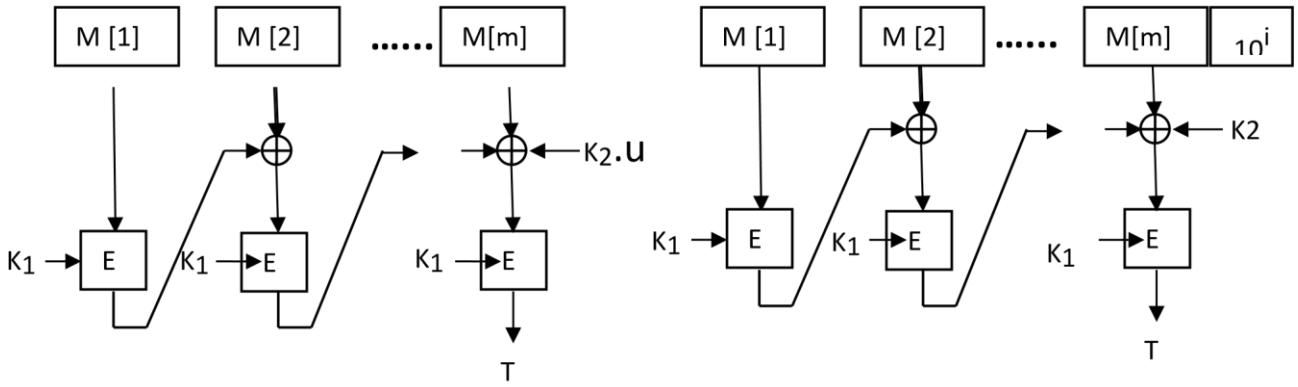

Figure 7 :TMAC

3.3.5 OMAC 1

Only one key, K (k bits) of a block cipher E, is used in one-key CBC MAC. Previously, XCBC required three keys, totaling (k+2n) bits, while TMAC required two keys, totaling (k+n) bits, where n denotes E's block length. OMAC1 and OMAC2 are known by the generic term OMAC. OMAC1 is created by substituting (K_2, K_3) in GF (2^n) with (L.u, L.u2) for some non-zero constant u, where L is defined by

$$L = E_K (0^n).$$

OMAC2 is similarly obtained by using (L.u, L.u^{-1}).

Algorithm

INPUT: block of message (M) *OUTPUT:*

MAC (Tag)

Algorithm OMAC-family$_k$(M)

$L = E_K(Cst)$

$Y[0] = 0^n$

Partition M into M [1] ……M[m]

for i = 0 to m-1 do

$Y[i] = E_{K1}(Y[i-1] \oplus M[i])$

$X[m] = pad_n(M[m] \oplus Y[m-1])$ if | M[m] | = n then $X[m] = X[m] \oplus L.u$

else $X[m] = X[m] \oplus L.u^2$

$T = E_K(X[m])$ return

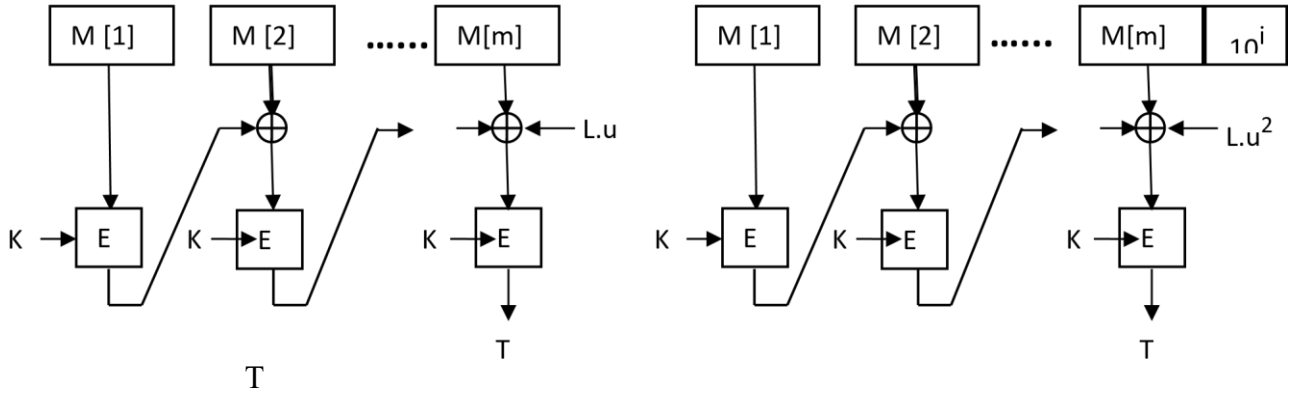

Figure 8: OMAC1

4. Experimental Setup

The experiments have been conducted on a laptop with an Intel Core i5 -2450M Processor 2.50GHz having 4 GB RAM. The operating system is Windows 7 Ultimate running in 64-bit mode. The system is running the java VM 22.0 –b10, Java HotSpot (TM) 64-Bit Server 1.7.0_02 with NetBeans IDE 7.0.1.

4.1 Measuring Cost

There is some extra cost which may be added to the absolute cost for creating message authentication code using different CBC MAC algorithm and it's all variants but this is equally affected to all candidate's algorithm on the execution. The system time in nanosecond is taken just before the execution of code segment for generating message authentication code (MAC) or tag along with key generation time in each algorithm and the completion of the execution. The time spending for creating MAC is calculated by subtracting start time taken before execution from completion time taken after completing execution of specific code segment which is used to produce MAC. The time required for each algorithm is calculated as follows:

```
long     startTime = System.nanoTime();
// CreateMAC function call long        timeRequired =
System.nanoTime() -  startTime;
```

Various processes may be run in background of system so absolute measurement may not be measure due to this reason, time needed for creating MAC in all algorithm may not be observe same in every run of program. Therefore at least 5 times, the program implemented in java is run in architecture describe as above section and finally average required time observed in every run is calculated as:

Average required Time = $\frac{1}{5}\sum_{i=1}^{5} T_i$ where T_i represent time obtained in i^{th} run of execution.

This average calculated time is used to calculate cycle per byte.

4.2 Measuring Performance

Timing cryptographic primitive is useful when analyzing the performance of multiple algorithms on a single machine but it may vary on other machine therefore, cryptographer prefer to measure how many cycles it takes to process each byte. Different cycle/byte is calculated in the same box also because of background other process. So, to optimize such extra cost, average is taken running multiple times in same machine for each candidate algorithms.

In this paper Cycle/byte calculation with the following parameters: time in second spent creating MAC(T_s), frequency of the CPU in Hz(F) and message input in bytes(L). the formula for creating cycle/byte suggested by (Stallings, 2010) is: Cycle/byte $= \frac{T_s * F}{L}$

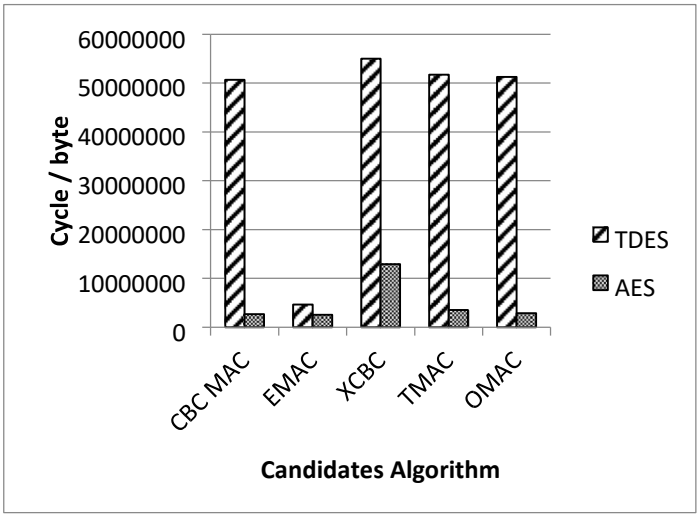

Figure 9: Performance of CBC MAC with its variants for small message size (29 byte) with encryption algorithm AES and TDES.

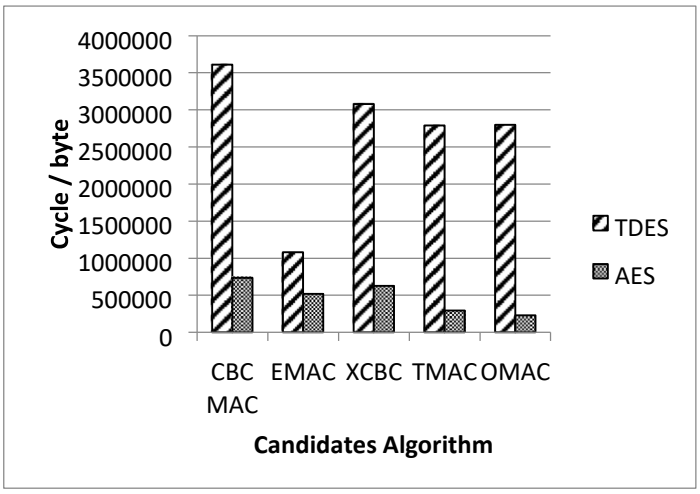

Figure 10: Performance of CBC MAC with its variants for small message size (595 byte) with encryption algorithm AES and TDES.

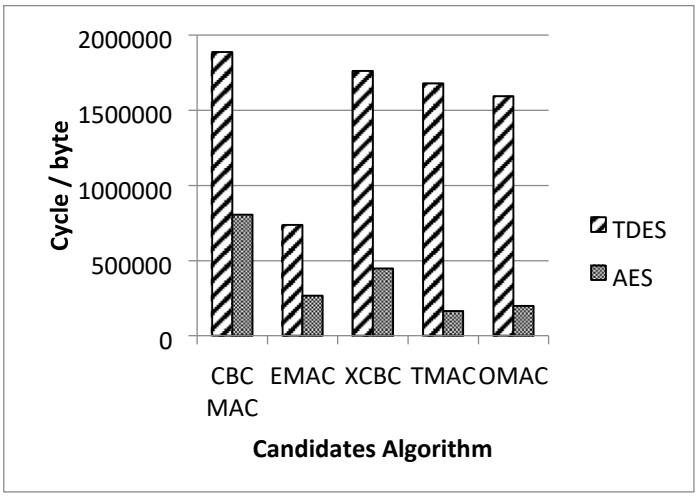

Figure 11: Performance of CBC MAC with its variants for message size (1KB) with encryption algorithm AES and TDES.

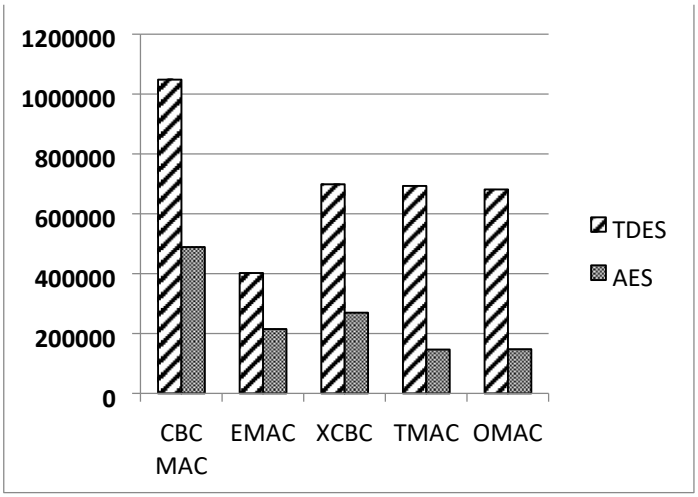

Figure 12: Performance of CBC MAC with its variants for message size (2KB) with encryption algorithm AES and TDES.

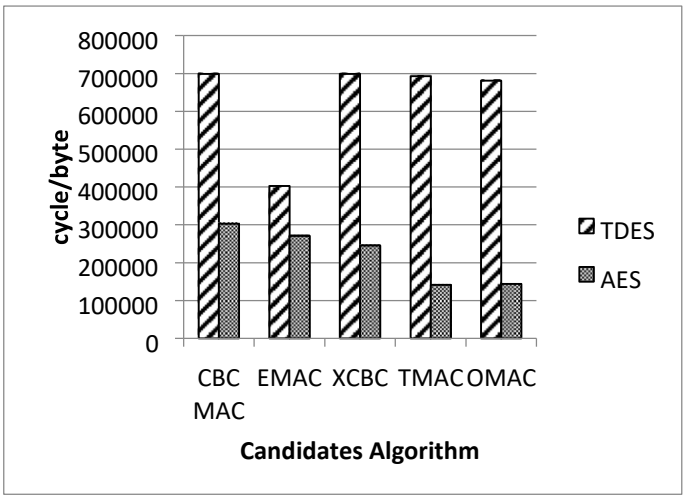

Figure 13: Performance of CBC MAC with its variants for small message size (5KB) with encryption algorithm AES and TDES

5. Results

If the encryption algorithm is TDES, Encrypted MAC (EMAC) seems to be the better cryptographic variants of CBC MAC whatever the size of message is taken to generate message authentication code (MAC). But, the nature of time required to create MAC changed according to input size of message in the case of AES. For small size of message, EMAC seems to be best model in AES algorithm also but it does not remain same when input size increased. By this way, Two-key MAC (TMAC) seems to the best cryptographic message authentication code algorithm based on cipher blocking chaining (CBC) mode. It is observed that TMAC yields 3%80% better performance (cycle/byte) than other variants when encryption algorithm is AES for small size message when size is increased it yields to 2%-53%. EMAC yields 90%-91% better performance than other variants when encryption algorithm is TDES for small size message when size is increased it yields to 40%-42%. For comparing two symmetric encryption algorithms, TDES and AES, it seems AES algorithm is best to use for every variant of CBC MAC. TDES has 5-11 times higher cycle/byte than AES.

6. Conclusion

The variants of cipher block chaining-based message authentication code algorithm are discussed along by using their java implementation. The result of empirical performance comparison shows that two variants of CBC MAC perform better depending on the message size and symmetric encryption algorithm. First EMAC or Encrypted MAC needs fewer cycles for processer to process byte when symmetric encryption algorithm is triple DES. In this encryption algorithm EMAC remains same performance by comparing with other variants but in case of AES result is not same for all size of message. For smaller size EMAC shows better performance than other when size of message is increased, TMAC shows the better performance. Comparatively OMAC is also shows better performance when large size of message is used to create MAC than other variants like CBC MAC, XCBC, and EMAC if the encryption algorithm is AES. For TDES for every size of input to the algorithm, EMAC shows better performance.

References

Iwata, T., & Kurosawa, K. (2003). *TMAC: Two-key CBC MAC.* CT-RSA 2003. LNCS, vol. 2612, pp. 33–49. Springer, Heidelberg.

John, B., & Phillip, R. (2000). *CBC MACs for arbitrary-length messages: The three key constructions.* Springer- Verlag.

Kurosawa, K., & Iwata, T. (2003/082). *Stronger security bounds for OMAC, TMAC and XCBC.* ePrint Archive.

Kurosawa, T. I. (2003). One-Key CBC MAC. Pre-proceedings of Fast Software Encryption. *OMAC:*.

Petrank, E., & Rackoff, C. (2000). CBC MAC for real-time data sources. *J.Cryptology, vol. 13, no. 3, pp.315–338, Springer-Verlag*.

Rogaway, P. (2000). PMAC: A Parallelizable Message Authentication Code.

Sarkar, P. (2009). Pseudo-Random Functions and Parallelizable Modes of Operations of a Block Cipher.

Stallings, W. (2010). *Cryptography And Network Security Principles and Practice , Prentice all , Fifth Edition.*