Available online at http://acfa.apeejay.edu

APEEJAY JOURNAL OF COMPUTER SCIENCE AND APPLICATIONS

ISSN:0974-5742(P)
Vol. 3, January 2015

PMAC: A Fully Parallelizable MAC Algorithm

Neeru Mago
Department of Computer Science and Applications
Panjab University Swami Sarvanand Giri Regional Centre, Hoshiarpur
neerumago80@gmail.com

ABSTRACT
In many applications of cryptography, assuring the authenticity of communications is as important as protecting their secrecy. The main goal of cryptography is to preserve the integrity of messages exchanged over public channels. A message authentication code algorithm (MAC) is designed for the sole purpose of preserving message integrity [1].A well known and secure method of providing message authentication is to compute a Message Authentication Code (MAC), often called a cryptographic checksum, on the message. Information security, including integrity and privacy, is an important concern among today's computer users due to increased connectivity. In this paper, we will describe a MAC which is deterministic, parallelizable, and uses only |M|/n block-cipher invocations to MAC a non-empty string M (where n is the block size of the underlying block cipher), known as PMAC, its characteristics and various algorithms.
Keywords : *Encryption, MAC, Hash Function, Security, PMAC.*

1. INTRODUCTION

Authentication, which certifies data integrity and data origin, is becoming an important technique because the transfer of valuable information needed for electronic funds transfer, business contracts, etc. must be made across computer networks. Data integrity ensures that the data has not been modified or destroyed. Data origin authentication is the verification that the source of data received is as claimed. Message authentication allows one party (the sender) to send a message to another party (the receiver) in such a way that if the message is modified, then the receiver will almost certainly detect this. Message authentication is said to protect the integrity of a message, ensuring that each message that it is received and deemed acceptable is arriving in the same condition that it was sent out with no bits inserted, missing, or modified.
Message Authentication is concerned with: protecting the integrity of a message, validating identity of originator, & non-repudiation of origin (dispute resolution). There are **three types of functions** [9] that may be used to produce an authenticator:

- message encryption,
- message authentication code (MAC),
- hash function.

Message encryption by itself can provide a measure of authentication. Here, the ciphertext of the entire message serves as its authenticator, on the basis that only those who know the appropriate keys could have validly encrypted the message. With public-key techniques, can get a digital signature which can only have been created by key owner.

Message authentication code (MAC): An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K. A MAC function is similar to encryption, except that the MAC algorithm need not be reversible, as it must for decryption.

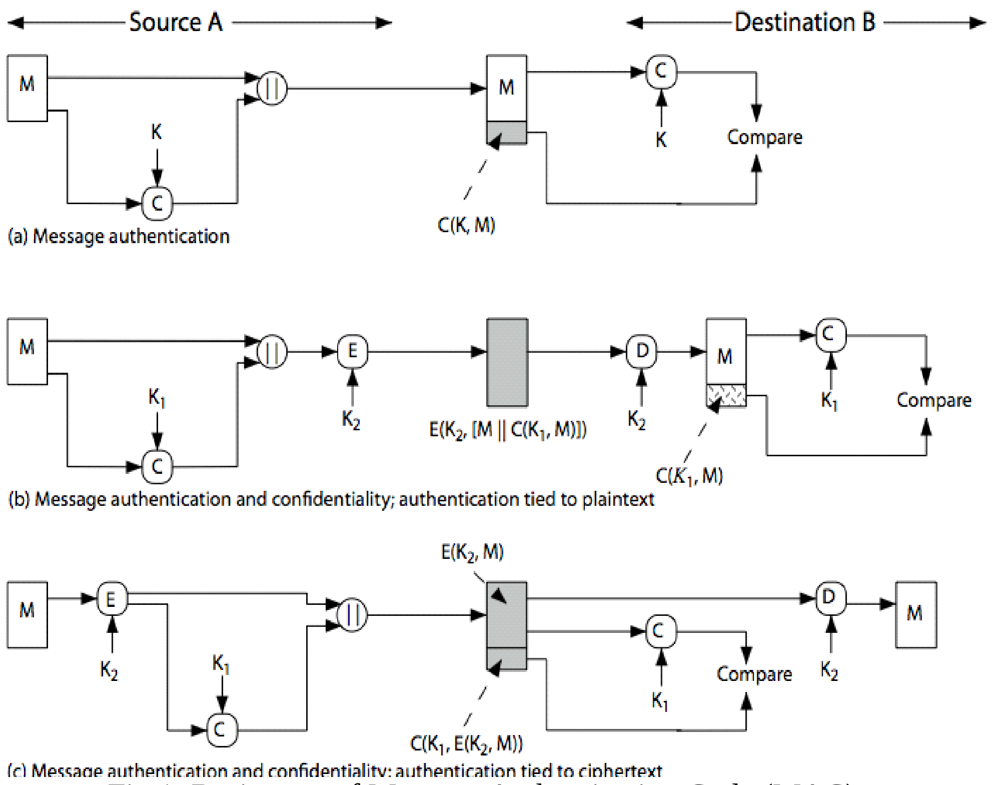
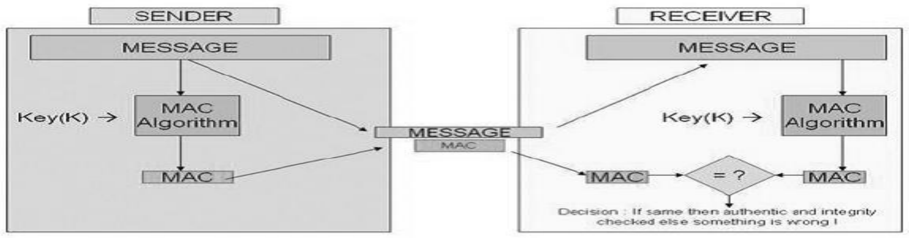

Source A
Destination B
M
K
C
||
M
C
K
Compare
(a) Message authentication
C(K, M)
M
K_1
C
||
E
K_2
$E(K_2, [M || C(K_1, M)])$
D
K_2
M
C
K_1
Compare
$C(K_1, M)$
(b) Message authentication and confidentiality; authentication tied to plaintext
$E(K_2, M)$
M
E
K_2
K_1
C
||
C
K_1
Compare
D
K_2
M
$C(K_1, E(K_2, M))$
(c) Message authentication and confidentiality; authentication tied to ciphertext

Fig 1: Basic uses of Message Authentication Code (MAC)

A message authentication code (MAC) function computes a MAC from a message and a secret key. If the originator and the receiver share knowledge of that secret key, the receiver can calculate the same function of the message and secret key and see if it matches the MAC accompanying the message. If the MAC matches, then the receiver knows, within the strength of the MAC function and key, that somebody with possession of the secret key produced the MAC. Of course, every receiver that can verify the MAC needs to know this secret key. Thus all the holders of that secret key can create valid MACs even if they should only receive and verify these codes. So a MAC is a symmetric-key method to ensure data integrity and authenticity.
A difficulty with MAC authentication in a system with multiple originators and receivers is that we must choose between two strategies, both of which have problems:
1. We could have a different secret for every pair of entities. This method is logistically difficult because the number of keys increases with the square of the number of entities and the keys must be securely distributed. If the system includes E number of entities, we should have E(E − 1)/2 secret keys.
2. Share one secret among all the entities. This technique is relatively insecure. The more entities that have a secret, the more likely the secret is to be compromised due to loss, subversion, or betrayal. This technique also means the same secret will be used many times; the more exposures of the uses of a secret, the easier an adversary may find it to break that secret analytically. In addition, with this strategy any of the entities can forge messages from any of the other entities and a recipient will be unable to detect this fraud based on the MAC.
MACs are usually implemented through keyed hash functions. Usually a MAC is a public algorithm with a secret compression function. In other words the secret key determines which compression function we should use among a family of functions. The family can be a family of random functions or a family of random permutations.



Hash function: A variation on the message authentication code is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

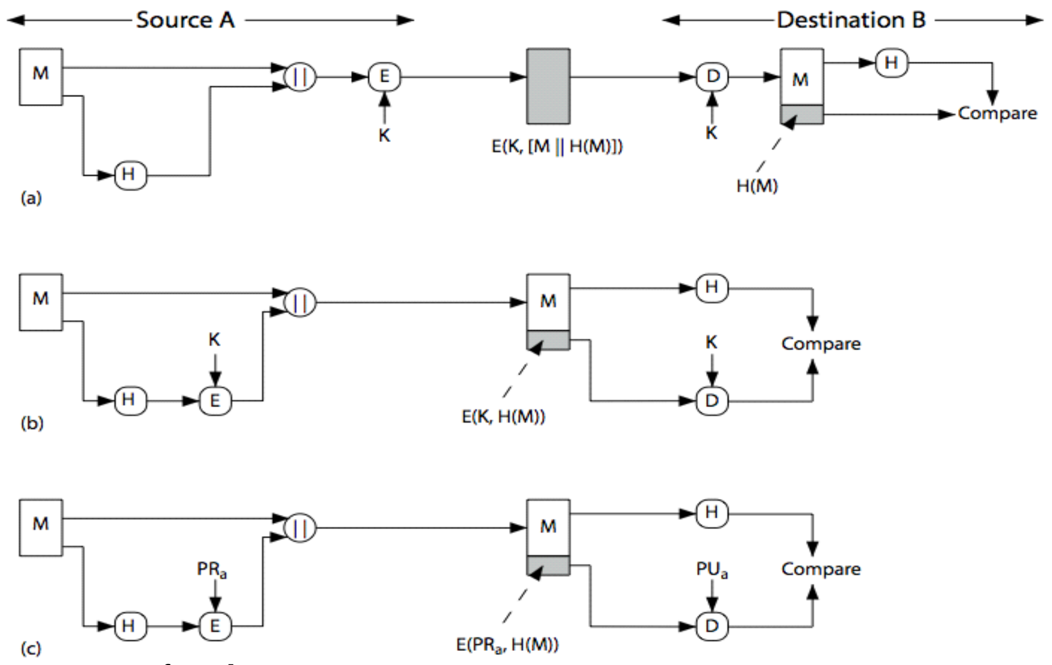
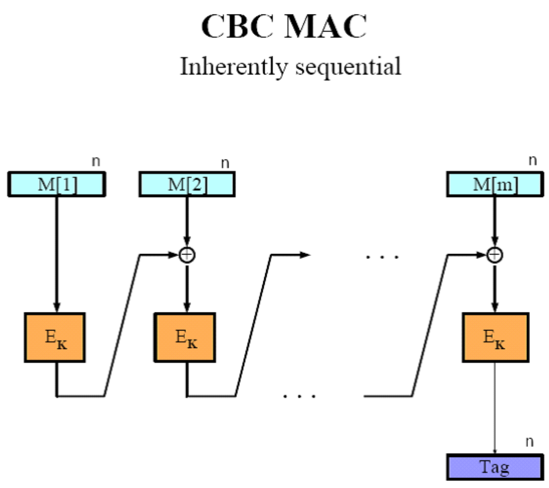


Fig3: Basic uses of Hash Functions

2. SURVEY ON VARIOUS MAC ALGORITHMS

There are various MAC algorithms that are used in practice. The most well known way to MAC uses a block cipher is called the **CBC MAC** [3]. It's defined, for example, in ISO 9797, ANSI X9.9, and ANSI X9.19. The message M must have a number of bits which is a positive multiple of the block length (128 bits for AES). The message is regarded as a sequence of 128-bit words, M=M[1]M[2]...M[m]. The MAC, let us write CBC(K,M), is defined as C[m], where C[i]=AES(K,C[i-1] xor M[i]) for i>0 and and C[0]=ZERO (the block of 128 zero-bits).

The basic CBC MAC is usually "embellished" a bit. **First**, the domain is extended to arbitrary bits strings, usually by padding with a 1-bit and then the minimal number of 0-bits to reach a multiple of 128 bits. **Second**, a particular security problem is fixed. The basic CBC MAC works correctly when all the strings you want to MAC have one fixed length (this is a well-known result of [Bellare, Kilian, and Rogaway]) but it does not work correctly when message lengths may vary. To fix this problem extra processing is often done to C[m] such as enciphering it under a separate key. **Third**, many standards allow you to take a prefix of the full n-bit MAC (where n is the block length). The three changes above represent one way to fix up the basic CBC MAC. A more recent suggestion due to [Black, Rogaway] more efficiently deals with padding and length-variability.



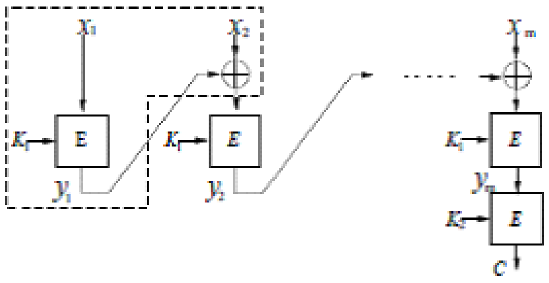
There are **several variants of CBC-MAC**.

DAC:Data Authentication Algorithm (DAA) is a former U.S. government standard for producing cryptographic message authentication codes. According to the standard, a code produced by the DAA is called a Data Authentication Code (DAC). The algorithm is not considered secure by today's standards. The DAA is equivalent to CBCMAC, with DES as the underlying cipher, truncated to between 24 and 56 bits (inclusive).

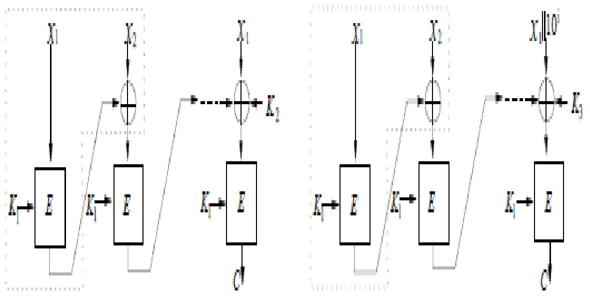
CMAC : Cipher-based MAC (CMAC) is a block cipher-based message authentication code algorithm, it may be used to provide assurance of the authenticity and, hence, the integrity of binary data. This mode of operation fixes security deficiencies of CBC-MAC (CBC-MAC is secure only for fixed-length messages). The core of the CMAC algorithm is a variation of CBC-MAC that Black and Rogaway proposed and analyzed under the name XCBC.

OMAC(one key MAC): OMAC allows and is secure for messages of any bit length (while the CBC MAC is only secure on messages of one fixed length, and the length must be a multiple of the block length). Officially there are two OMAC algorithms (OMAC1 and OMAC2). OMAC1 is equivalent to CMAC. NIST Special Publication 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication has been finalized on May 18, 2005.

EMAC: The Encrypted MAC (EMAC), also known as double MAC (DMAC), is a popular variant of the CBC-MAC developed by the RACE project. It is derived from the CBC function by additionally encrypting the output with an independent permutation and secure without any restriction on the message space. The EMAC is also one of the message authentication codes recommended by NESSIE. EMAC is the first encrypted MAC . It is obtained by encrypting the CBC-MAC value by E again with a new key K2. That is, EMACK1,K2 (M) = EK2 (CBCK1 (M)), where K1 is the key of the CBC-MAC and CBCK1 (M) is the CBC-MAC value of M.



XCBC: The XCBC scheme was originally proposed by Black and Rogaway in 2000, with the objective of providing a provably secure CBC-MAC scheme which minimizes the number of block cipher encryptions and decryptions. In general, XCBC takes three keys: one block cipher key K1, and two n-bit keys K2 and K3. If the message length is the multiple of n, let K = K2, P = M. Otherwise, let K = K3, P = M||10i, where i = n − 1 − (|M| mod n). Write P = x_0 ||x_1 || • • • ||x_t , y_0 = 0. Then yi = E_{K1} (x_i xor y_{i-1}), for i = 1, . . . , t − 1. The XCBC value is C = E_{K1} (x_t xor y_{t-1} xor K).



RMAC: It was proposed by Jaulmes, Joux and Valette, which is an extension of EMAC. The block cipher algorithms currently approved to be used in RMAC are the AES and triple- DES.

TMAC: stands for **two-key MAC**. It is a refinement of XCBC shown by Black and Rogaway. It was proposed by Kurosawa and Iwata with the goal of reducing the number of required keys from three to two.
UMAC(Universal MAC): A message authentication code based on universal hashing (UMAC), is a type of message authentication code (MAC) calculated choosing a hash function from a class of hash functions according to some secret (random) process and applying it to the message. The resulting digest or fingerprint is then encrypted to hide the identity of the hash function used. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message. A UMAC has provable cryptographic strength and is usually a lot less computationally intensive than other MACs.

HMAC(Hashed MAC) : A keyed-hash message authentication code (HMAC), is a type of message authentication code (MAC) calculated using a cryptographic hash function in combination with a secret key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message. Any iterative cryptographic hash function, such as MD5 or SHA-1, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-MD5 or HMAC-SHA-1 accordingly. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function, on the size and quality of the key and the size of the hash output length in bits.

PMAC (Parallelizable MAC): PMAC, a fully parallelizable MAC scheme based on block cipher, is proposed by Black and Rogaway in Eurocrypt 2002. All the incoming data blocks are passed through block ciphers parallelly, essentially reducing the processing time.

3. PMAC

Many popular message authentication codes (MACs), like the CBC MAC and HMAC, are inherently sequential: one cannot process the i-th message block until all previous message blocks have been processed. This serial bottleneck becomes increasingly an issue as commodity processors offer up more and more parallelism, and as increases in network speeds outpace increases in the speed of cryptographic hardware. By now there would seem to be a significant interest in having a parallelizable MAC which performs well in both hardware and software, built from a block cipher like AES. And modern processors which offer up lots of registers and instruction-dispatch units will be limited by the amount of parallelism there is to exploit in the underlying block cipher. PMAC was designed to address this issue, being fully parallelizable. In saying that PMAC is *fully* parallelizable we mean that you are effectively unlimited in how much parallelism you can extract. PMAC was proposed in response to NIST's call for contributions for a first modes-of-operation workshop. Earlier versions of this writeup were submitted to NIST and posted to their website (Oct 2000, Apr 2001).

3.1 PMAC's PROPERTIES

PMAC has been designed to simultaneously have lots of desirable properties. These include the following.

- PMAC is actually more than a MAC: its a *pseudorandom function*. Roughly said, this means that if K is random and unknown to the adversary, $f(x)$=PMAC(K,x) will look like a random function.
- PMAC is fully parallelizable.
- PMAC works on any bit string.
- PMAC works with any block cipher.
- PMAC provides a tag with as many bits as you want, up to the block length n of the underlying block cipher.
- PMAC makes an optimal number of block-cipher calls for a conventional mode of operation, namely, roundup(|M|/n).
- PMAC is deterministic: it uses no counter/IV/nonce and no random bits. This has been true in all traditional MACs.
- PMAC uses only a single block-cipher key, and every invocation of the block cipher is keyed by that one key.
- PMAC only use the forward direction of the underlying block cipher.

- PMAC key setup is very cheap: one block-cipher call and, if desired, a few XORs and conditional (128-bit) shifts.
- PMAC generates a sequence of offsets, which it does in a very cheap way. Each offset is computed from the previous one either by xoring the previous offset by a value looked up in a small table or by doing a few shifts and xors.
- PMAC is nearly endian-neutral: the scheme can be implemented just as fast on a big-endian machine or on a little-endian machine.
- PMAC doesn't need much memory to run.
- PMAC avoids 128-bit addition (which is endian-biased and can be expensive in software or dedicated hardware). It uses xors instead.
- PMAC is incremental.
- PMAC is simple to understand and implement. One doesn't have to understand the mathematics to implement the scheme.
- PMAC is provably secure. It provably meets its goals, as long as the underlying block ciphers meet a standard cryptographic assumption.

3.2 The PMAC Algorithm / Working

Addition, multiplication, and Final(.) We assume an addition operator + from $\{0,1\}^n * \{0,1\}^n$ to $\{0,1\}^n$ and a multiplication operator (with no explicitly written symbol) from {1, 2, 3,…..} * $\{0,1\}^n$ to $\{0,1\}^n$. We also assume a map Final : $\{0,1\}^n \rightarrow \{0,1\}^n$.

3.2.1 PMAC/add

For the addition modulo 2^n version of PMAC, PMAC/add, instantiate + by computer addition of n-bit words (ignoring any carry) and instantiate iL, for i >= 1, by repeated addition. Let Final(L) be L', the bitwise complement of L. Given a k-bit key K, derive from it a key L by way of $L = E_K(0^n) \vee 0^{n-1}1$. This ensures that L is odd.

Definition of PMAC We now define PMAC. When addition and multiplication are as just given, we are defining PMAC/add. Given a string M, its MAC is computed as specified below.

```
Algorithm PMAC
Let m= max { 1, |M|/n}
Let M[1],……….M[m] be strings s.t. M[1]………M[m] =M and |M[i]| =n for 1<= I < m
For i= 1 to m-1 do
        C[i] = $E_k$ (M[i] + iL)
If  |M[m]|= n then
        preTag = C[1] XOR C[2] XOR………….XOR C[m-1] XOR M[m]
        Tag = $E_k$ (preTag +Final(L))
Else
        W = $pad_n$ (M[m])
        preTag = C[1] XOR C[2] XOR………….XOR C[m-1] XOR W
        Tag = $E_k$ (preTag)
T = Tag[ bits 1to tagLen]
Return T
```

As the MAC is deterministic, a separate MAC verification algorithm need not be given: the algorithm is to compute the MAC that should accompany the message, and see if it matches the MAC received.

3.2.2 PMAC/mod p

This section gives a slight variant of PMAC. A better security bound for PMAC can be obtained by computing iL modulo p, instead of computing iL modulo 2^n. When addition/multiplication is defined under this revised semantics, L need not be odd; select $L = E_K(0^n)$ instead. Multiplication by a positive number i means repeated addition.

3.2.3 PMAC/xor

In this section we describe yet another method of offsetting the blocks M[1],M[2], ……,M[m - 1]. Assume we are using **AES** (with whatever key length you want). ***L*** is a 128-bit string where *L*<<1 for the left shift of *L* by 1 bit (where the first bit vanishes and a 0 comes into the last bit). Write *L*>>1 for the right shift of *L* by 1 bit (where the last bit vanishes and a 0 comes into the first bit). For a nonzero number *i*, let **ntz(*i*)** be the number of trailing 0-bits in the binary representation of *i* (so, for example, ntz(52)=2, because 52 is 110100 in binary, which ends in two zeros). Where *x* is a string of at most 128 bits, let pad(*x*) be *x* if *x* has exactly 128 bits, and let it be *x* followed by a 1-bit followed by 127-|x| 0-bits otherwise.

Let *M* be the message you want to MAC, and let *K* be the PMAC key, which is just an AES key. Let *L* be AES(*K*, ZERO). Let *L*(0) be *L* and, for *i*>0, let *L*(*i*) be *L*<<1 if the first bit of *L*(*i*-1) is 0, and let *L*(*i*) be(*L*(*i*-1)<<1)**xor** hex87 otherwise. Let *L*(-1) be *L*>>1 if the last bit of *L* is 0, and let *L*(-1) be *L*>>1 xor hex43 otherwise. Break the message *M* into *m*≥1 blocks *M*[1]*M*[2]...*M*[*m*] where each block except the last one has 128 bits. The last one may have fewer than 128 bits. Now we MAC as follows.

```
Algorithm PMAC $_K$ (M)
L(0) =$E_K$ (0)
L(-1) =lsb (L(0)) ? (L(0) >> 1) XOR Const 43 : (L(0) >> 1)
For I = 1,2,……….do
        L(i) = msb (L(i-1)) ? (L(i-1) << 1) XOR Const 87 : (L(i-1) << 1)
Partition M into M[1]………..M[m]
Offset = 0
For I = 1 to m-1 do
        Offset = Offset XOR L(ntz (i))
        $\sum = \sum$ XOR $E_K$ (M[i] XOR Offset)
$\sum = \sum$ XOR pad(M[m])
If  |M[m]| =n then
        $\sum = \sum$ XOR L[-1]
Fulltag = $E_K(\sum)$
Tag = first t bits of Fulltag
Return tag
```

Unlike customary modes for message authentication, the construction here is fully parallelizable. This will result in faster authentication in a variety of settings.

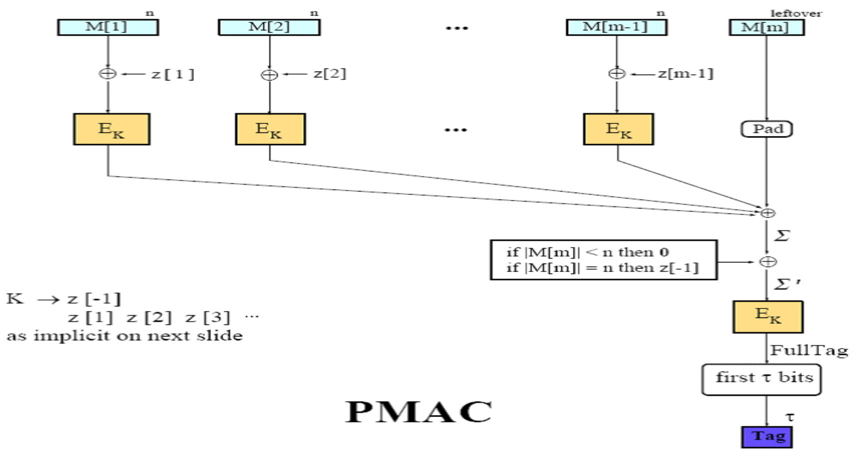


TABLE 1: Comparative Study on Various PMAC algorithms

Scheme	Meaning of A+B	Meaning of iL, for i>= 1	Meaning of Final (L)	Definition of L
PMAC/add	Add 128 bit numbers. Ignore any carry	Repeated addition.	L'	$E_K(0^{128})$ V 1
PMAC/mod	Add 128 bit numbers mod p.	Repeated addition.	L'	$E_K(0^{128})$
PMAC/xor	XOR	Multiply λ(i) by L in GF (2^{128}), where λ(i) is the ith word in canonical Gray-code ordering	L(127), which is 2^{127} * L, this arithmetic in GF (2^{128})	$E_K(0^{128})$ Λ Const

4. ANALYSIS

Unlike the XOR MAC, new algorithm, PMAC, doesn't waste any block-cipher invocations because of block-indices. We optimally deal with short final blocks; we correctly MAC messages of arbitrary and varying bit lengths. The result is that PMAC makes do with just $|M|/n$ block-cipher calls to MAC a non-empty message M using an n-bit block cipher. PMAC is deterministic, freeing the user from having to provide a counter or random value, and making the MAC shorter. Overhead beyond the block-cipher calls has been aggressively optimized, so a serial implementation of PMAC runs just a few percent slower than the CBC MAC. Besides the efficiency measures already mentioned, PMAC uses very little key-setup: one block cipher call. (A few shifts and conditional xors are also used.) The PMAC key is a single key for the underlying blocks cipher; in particular, we forgo the need for key-separation techniques. Avoiding multiple block-cipher keys saves time because many block ciphers have significant key-setup costs.
We prove PMAC secure, in the sense of reduction-based cryptography. Specifically, we prove that PMAC approximates a random function (and is therefore a good MAC) as long as the underlying block cipher approximates a random permutation. The actual results are quantitative; the security analysis is in the concrete-security paradigm.

4.1 Security of PMAC

One of the most important requirements of MAC is that given a massage M and a l k-bit secret key K, the computation of the MAC value $MAC_K(M)$ should be easy. However, it should be computationally infeasible to find $MAC_K(M)$ without knowing K. **Security properties** of MAC function include:

Existential Forgery: An adversary is able, without initial knowledge of K, to get a corresponding MAC C for any message M, which has not been MACed by the legitimate MAC generator. The message M may not have any particular meaning.

Selective Forgery: An adversary is able to determine the MAC for a message of his choice.

Second Preimage Resistance: Second preimage resistance is sometimes referred to as weak collision resistance. If an adversary observes M and the corresponding MAC C, constructing a message M′ != M with MAC_K (M′) = C should be computationally infeasible for him. The relation $Pr[MAC_K(M′) = C] = 2^{-lm}$ should hold in this case, where lm is the length of the tag.

Universal forgery: An adversary is able to find a MAC for every given message. This attack is much more powerful than previous cases.

Key Recovery Attack: A key recovery attack is more devastating than forgery. In this case an adversary is able to recover K itself, and thus can perform arbitrary forgeries.

5. FUTURE WORK

Despite a number of secure algorithms that have been proposed, the trade-offs made between security and performance demands further research toward improvement. For example, in bulk data transfer, especially in large messages, the secured processing time takes much longer than non-secured processes. This is due to crypto operations, which include symmetric encryption operations and hashing functions. In the current bulk data transfer phase in Secure Socket Layer (SSL), the server or the client firstly calculates the Message

Authentication Code (MAC) of the data, and then performs the symmetric encryption on the data together with the MAC.

CONCLUSION

We define and analyze a simple and fully parallelizable block-cipher mode of operation for message authentication. Parallelizability does not come at the expense of serial efficiency: in a conventional, serial environment, the algorithm's speed is within a few percent of the (inherently sequential) CBC MAC. The new mode, PMAC, is deterministic, resembles a standard mode of operation (and not a Carter-Wegman MAC), works for strings of any bit length, employs a single block-cipher key. We prove PMAC secure, quantifying an adversary's forgery probability in terms of the quality of the block cipher as a pseudorandom permutation.

REFERENCES

[1] D.Ganesh, A.Jahnavi, M.Trivikram, "Dynamic Message Authentication Code for Short Messages", International Journal of Computer Science and Information Technologies, Vol. 6 (2) , 2015, 1234-1238, ISSN: 0975-9646.

[2] N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, and T. Kohno, "Helix: Fast encryption and authentication in a single cryptographic primitive," in Proceedings of Fast Software Encryption–FSE'03, vol. 2887, Lecture notes in computer science. Springer, 2003, pp.330–346.

[3] B. Alomair and R. Poovendran, "E-MACs: Towards More Secure and More Efficient Constructions of Secure Channels," in the 13th International Conference on Information Security and Cryptology – ICISC'10. Springer, 2010.

[4] PMAC: A Parallelizable Message Authentication Code Phillip Rogaway University of California at Davis (USA) and Chiang Mai University (Thailand), October 16, 2000.

[5] Hashem Mohammed Alaidaros, Mohamed Othman and Mohd Fadlee A. Rasid, "Improving Security Performance with Parallel Crypto Operations in SSL Bulk Data Transfer", International Journal of Cryptology Research 1(2): 235-243 (2009).

[6] J. Black, P. Rogaway, "A Block-Cipher Mode of Operation for Parallelizable Message Authentication", Advances in Cryptology, February 15, 2002.

[7] Mridul Nandi and Avradip Mandal, "Improved Security Analysis of PMAC", Crypt. 1 (2007), 1–14 DOI 10.1515 / JMC.2007.

[8] Ioannis Yiakoumis, Markos Papadonikolakis, Harris Michail, Athanasios P. Kakarountas "Efficient Small-Sized Implementation of the Keyed-Hash Message Authentication Code", Serbia & Montenegro, Belgrade, November 22-24, 2005.

[9] Chapter 11: Message Authentication and Hash Functions. Fourth Edition by William Stallings.

[10] Nitesh Saxena, Nakul Sharma, Karthik Erapalli, "Hash Functions and Message Authentication", March 27, 2008.

[11] MAC Constructions: Security Bounds And Distinguishing Attacks**, http://hdl.handle.net/10012/3058**

[12] Kazuo Ohta1 and Mitsuru Matsui, "Differential Attack on Message Authentication Codes".

[13] T. Iwata and K. Kurosawa, "omac: One-key cbc mac," in Fast Software Encryption–FSE'03, vol. 2887, Lecture notes in computer science. Springer, 2003, pp. 129–153.

[14] M. Bellare, R. Guerin, and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions," in Advances in Cryptology–CRYPTO'95, vol. 963, Lecture Notes in Computer Science. Springer, 1995, pp. 15–28.