A Privacy Preserving Message Authentication Code

Dang Hai Van, Nguyen Dinh Thuc
University of Science, VNU-HCM, Vietnam
{dhvan,ndthuc}@fit.hcmus.edu.vn

***Abstract*—In this paper, we propose a new message authentication code which can preserve privacy of data. The proposed mechanism supports to verify data integrity from only partly information of the original data. In addition, it is proved to be chosen-message-attack secure and privacy-preserving. We also conduct an experiment to compare its computation cost with a hash message authentication code.**

***Index Terms*—MAC, keyed hash function, data integrity, data privacy, data audit**

I. Introduction

Maintaining integrity is about detecting if an adversary has tampered with the data sent by one to another. Most common methods of integrity checking are based on using a shared key and they are called Message Authentication Code or MAC. In this setting, the sender sends data with an appended tag which is computed as a function of the data and the shared key. The recipient then re-computes the tag from the received data and compares it with the received tag. If both match together, it means that the data is valid. Otherwise, the data has been interfered with. Message Authentication Code may be constructed from block cipher like DES, for e.g. CBC MAC, or cryptographic hash function like SHA, for e.g. HMAC [1].

Up to date, when storage service in cloud like database-as-a-service (DaS) has became more common, integrity checking currently consists of identifying if a legitimate storage server has deleted or not disclosed loss or corruption of the data stored in it. Assuming that a legitimate server is believed not to use or reveal users data without permission. In case the stored data is huge, the data owner may decide to hire a third-party audit service provider which is called an auditor. The auditor takes charge of checking integrity of the data stored in the server. However, the data owner may not want to let the auditor know about the data (see [2], [3], [4], [5]). In such a case, both data integrity and privacy are important and the traditional MAC and HMAC could not resolve. It needs a mechanism to check integrity while preserving data privacy. Aimed to this motivation, we has developed a new message authentication code, called privacy preserving message authentication code or PPMAC.

II. Preliminaries

In this section we refer a brief description of Message Authentication Code and Chosen-message-attack security notion.

A. Message Authentication Code

Message Authentication Code (MAC) is the cryptographic primitive that is used for data integrity [6]. A MAC consists two algorithms:

- $\mathbf{Sign}$ - for signing or creating a tag for the message.
- $\mathbf{Ver}$ for verifying or checking if the message and the tag match together.

Denote

- k be the key shared between the data owner and the verifier, and n be its length.
- s be the tag of the message and t be its length.

Then the two algorithms $\mathbf{Sign}$ and $\mathbf{Ver}$ can be defined as functions

$\mathbf{Sign} : \{\mathbf{0},\mathbf{1}\}^{\mathbf{n}} \times \{\mathbf{0},\mathbf{1}\}^* \rightarrow \{\mathbf{0},\mathbf{1}\}^{\mathbf{t}}$ that maps a message of arbitrary length and the shared key of n bits into a signature or tag of t bits.

$\mathbf{Ver} : \{\mathbf{0},\mathbf{1}\}^{\mathbf{n}} \times \{\mathbf{0},\mathbf{1}\}^* \times \{\mathbf{0},\mathbf{1}\}^{\mathbf{t}} \rightarrow \{\mathbf{0},\mathbf{1}\}$ that maps a message, its tags and the shared key into one in values $\{0,1\}$. The return value 1 means that the tag matches to the message, and 0 otherwise.

In order to create the tag for the message x, the data owner computes

$$s = Sign(k,x) = Sign_k(x)$$

After that the data owner sends the pair of message and its tag or signature to the verifier. In order to audit the message, the verifier computes

$$Ver(k,x,s) = Sign_k(x,s)$$

The data is accepted as valid if the output $Ver_k(x,s) = 1$.

It is clear that in MAC, it requests the plain message as its input for verifying the tag. Therefore, MAC can't be used in the case of checking integrity while preserving data privacy as described in the section Introducton.

B. Chosen-message-attack secure MAC

A Message Authentication Code $(Sign, Ver)$ is said to be chosen-message-attack secure (CMA-secure) if for every n it is $(T(n),\epsilon(n))$-CMA-secure where $T(n),\epsilon(n)$ are polynomials. In other words, there is no polynomial-time adversary that succeeds in breaking it with polynomial probability. Below is the formal definition for a CMA-secure MAC.

Given two positive integers n, t and two functions
$\mathbf{Sign} : \{\mathbf{0},\mathbf{1}\}^{\mathbf{n}} \times \{\mathbf{0},\mathbf{1}\}^* \rightarrow \{\mathbf{0},\mathbf{1}\}^{\mathbf{t}}$,
$\mathbf{Ver} : \{\mathbf{0},\mathbf{1}\}^{\mathbf{n}} \times \{\mathbf{0},\mathbf{1}\}^* \times \{\mathbf{0},\mathbf{1}\}^{\mathbf{t}} \rightarrow \{\mathbf{0},\mathbf{1}\}$

Definition 1. *A pair of algorithms $(Sign, Ver)$ is a $(T(n),\epsilon(n))$-CMA-secure Message Authentication Code if:*

(1) ***Validity** For every message $x \in \{0,1\}^*$, shared key $k \in \{0,1\}^n$, $Ver_k(x, Sign_k(x)) = 1$*

978-1-4673-6537-6/15/$31.00 ©2015 IEEE

(2) ***Security*** *For any adversary limited up to time $T(n)$, consider the following experiment:*

- *Choose randomly a shared key k: $k \leftarrow_R \{0,1\}^n$.*
- *Let the adversary access to the two algorithms $Sign_k(.)$ and $Ver_k(.)$ as black boxes.*
- *The adversary outputs a pair of a message and a tag (x', s') where the adversary has never asked about the tag of x'. The adversary wins the experiment if (x', s') so that $Ver_k(x', s') = 1$.*

The probability that the adversary wins is at most $\epsilon(n)$.

III. A NEW MESSAGE AUTHENTICATION CODE

A. Privacy preserving message authentication code

Let us denote

- $h_k : \{0,1\}^* \to \{0,1\}^n$ be a cryptographic hash function, where k is a shared key,
- $H_i : \{0,1\}^z \to \{0,1\}^z$ be a family of one-way permutations
- $Q_k : \{0,1\}^* \to \{0,1\}^{n+1}$ be a prime generator which generates a $n+1$-bit prime number such that Q_k satisfies simultaneously 2 following conditions:
  (i) $Q_k(s) \neq Q_k(s')\forall s \neq s'$, and
  (ii) $Q_k(s) \neq Q_{k'}(s), \forall k \neq k'$.

  Note that it is not difficult to generate a random prime (see [7]).

And without loss the generality, suppose that

$$m = m_0||m_1||m_2 \in \{0,1\}^*$$

be a message, with $|m_0| = |m_1| = |m_2| = |m|/3 \cong z$.

Definition 2. ***A Privacy Preserving Message Authentication Code (PPMAC)***

Given $m = m_0||m_1||m_2 \in \{0,1\}^*$ be a massage. Let

$$\begin{aligned} M &= H_0(m_0) \oplus H_1(m_1) \oplus H_2(m_2), \\ f_i &= h_k(M \oplus H_i(m_i)), i = 0,1,2, \\ q_i &= Q_k(i||f_i), i = 0,1,2. \end{aligned}$$

Tthe privacy preserving message authentication code (PPMAC) of message m is defined as the solution of the system of congruences:

$$\begin{aligned} X &\equiv f_0 \pmod{q_0} \\ X &\equiv f_1 \pmod{q_1} \\ X &\equiv f_2 \pmod{q_2} \end{aligned}$$

The above PPMAC is a well-defined because X is the unique solution of the congruent system as it will be proved by the below theorem.

Theorem 3. *Given $m_i \in \{0,1\}^z, i = 0,1,2$.*
Let

- *$M = H_0(m_0) \oplus H_1(m_1) \oplus H_2(m_2)$, where $H_i, i = 0,1,2$, are permutations over $\{0,1\}^z$,*
- *$f_i = h_k(M \oplus H_i(M_i)), i = 0,1,2$, where h_k is a keyed hash functions,*
- *$q_i = Q_k(f_i), i = 0,1,2$ are $(n+1)$-bit primes.*

Then system

$$\begin{aligned} X &\equiv f_0 \pmod{q_0} \\ X &\equiv f_1 \pmod{q_1} \\ X &\equiv f_2 \pmod{q_2} \end{aligned}$$

has the unique solution in $Z_q, q = q_0 \times q_1 \times q_2$.

Proof. This theorem is a special case of the Chinese Remainder Theorem (see [8]). □

From Definition 2, we design a pair of algorithms for PPMAC as below.

PPMAC_Sign

(1) Compute
$M = H_0(m_0) \oplus H_1(m_1) \oplus H_2(m_2)$.
(2) Compute
$f_i = h_k(M \oplus H_i(m_i)), i = 0,1,2$.
(3) Generate
$q_i = Q_k(i||f_i), i = 0,1,2$.
(4) Solve system

$$\begin{aligned} X &\equiv f_0 \pmod{q_0} \\ X &\equiv f_1 \pmod{q_1} \\ X &\equiv f_2 \pmod{q_2} \end{aligned}$$

(5) Return X

PP_Ver

- Given $H_0(m_0) \oplus H_1(m_1), H_0(m_0) \oplus H_2(m_2)$ and the tag X

(1) Compute
$f_1 = h_k(H_0(m_0) \oplus H_2(m_2))$, and
$f_2 = h_k(H_0(m_0) \oplus H_2(m_1))$
(2) Compute
$q_i = Q_i(i||f_i), i = 1,2$
(3) Check if $X \pmod{q_i} = f_i, \forall i = 1,2$. If yes, return 1. Otherwise, return 0.

B. Security

Theorem 4. *If $\{h_k\}$ is a family of pseudorandom functions (PRF) then the defined PPMAC is a CMA-Secure MAC.*

Proof. Let A be an adversary that runs a chosen-message-attack against the PPMAC. At the end of the attack, the adversary comes up with a pair of message and its supposed tag (m', x'). The message m' must have never been asked for its tag before. Because $\{h_k\}$ is a family of pseudorandom functions, we can think that the adversary A chooses the tag x' by random. x' is the solution of the congruent system and x' is about $\mathcal{O}(n)$ bits. Therefore, the probability that the adversary

choose the correct x' is about $2^{-\mathcal{O}(n)}$. It deduces that the PPMAC is CMA-secure. □

Theorem 5. *The number of primes not exceeding x is asymptotic to $x/logx : \pi(x) \sim x/logx$.*

Proof. See [3], [4]. □

Corollary 6. *There are about $2^n(n-1)/(n(n+1))$ $(n+1)$-bit primes.*

Proof. By the Theorem 5,
- There are about $2^{n+1}/(n+1)$ bit primes.
- There are about $2^n/n$ bit primes.

Therefore the number of $(n+1)$-bit primes is

$$2^{n+1}/(n+1) - 2^n/n = (2^n(n-1))/(n(n+1))$$

□

Theorem 7. *The adversary who is not given the shared key cannot forge the PPMAC X*

Proof. Without the shared key and due to the number of primes as in Corollary 5, the adversary cannot generate or choose properly the primes primes $q_i, i = 0, 1, 2$, and generate $f_i, i = 0, 1, 2$. Therefore, it cannot establish a same congruent system to solve a same PPMAC X.

In addition, by By Theorem 4, the adversary cannot choose the PPMAC X properly. □

Lemma 8. *From $c_{ij} = a_i \oplus a_j, \forall 1 \leq i, j \leq n, i \neq j$, it is impossible to recover any $a_i \in \{a_1, ..., a_n\}$*

Proof. Indeed, without loss the generality, we consider the simplest example:

Given $a \oplus b$ and $b \oplus c$, it tries to recover an element in $\{a, b, c\}$

Denote that

$(1) = a \oplus b$, and

$(2) = b \oplus c$

We have,

$(3) = (1) \oplus (2) = (a \oplus b) \oplus (b \oplus c) = a \oplus c$

Similarly,

$(2) \oplus (3) = a \oplus b = (1)$

$(1) \oplus (3) = b \oplus c = (2)$

There is no way to discover a single element in $\{a, b, c\}$. □

Theorem 9. *The privacy of messages are preserved against the verifier.*

Proof. By Lemma 8, the verifier cannot find out any single value in $H_i(m_i), i = 0, 1, 2$. Even if the verifier knows a $H_i(m_i), i \in \{0, 1, 2\}$, it cannot recover the content of m_i because H_i is a one-way permutation. □

C. Comparison and Experiment

With other message authentication codes - MAC, it needs the original data and its tag or signature to check the data integrity. With our proposed message authentication code, PPMAC, it needs only a part of information of data. Indeed, assume that $m = m_0||m_1||m_2$ is a given message. To authenticate the data integrity, it needs only z-bit string $H_0(m_0) \oplus H_1(M_1)$ and z-bit string $H_0(M_0) \oplus H_2(M_2)$ where z is the bit length of $m_i, i = 0, 1, 2$. It does not need the whole original message m.

Moreover, the data content is protected against the verifier (Theorem 9). Due to its privacy preserving ability, PPMAC may require more computation than other MAC.

In addition, to compare computation cost of PPMAC and other MAC, we conduct an experiment using SAGE [1] to measure running time of PPMAC and a HMAC which is built upon the cryptographic hash function SHA256. The experiment is implemented in the system configuration as below:

TABLE I
SYSTEM CONFIGURATION

Processor: Intel Core i5-2430M CPU @ 2.4 GHz RAM: 4.00 GB Operating system: 32-bit Windows 7

Parameters and functions of PPMAC are chosen and implemented as

TABLE II
IMPLEMENTATION

Length of message (in bits): $48, 96, 144$ $H_i(i = 0, 1, 2)$ are defined using the miniAES in SAGE. h_k is defined using SHA256. The number of samples (the number of runs): 30 times

The result which are the average running time (in seconds) of PPMAC and HMAC are displayed in the following table

TABLE III
RUNNING TIME (IN SECONDS)

Message length (in bits)	PPMAC	HMAC
48	Sign: 8.013	Sign: 0.003
	Verify: 3.071	Verify: 0.003
96	Sign: 11.845	Sign: 0.003
	Verify: 3.096	Verify: 0.003
144	Sign: 16.287	Sign: 0.003
	Verify: 4.067	Verify: 0.003

From the result, it can be seen that the running time of PPMAC is longer then HMAC in both signing and verification process. It is natural because the computation cost required by

[1] http://www.sagemath.org/

PPMAC is more complicated. In addition, the PPMAC running time increases if the message length increases, while the running time of HMAC does not. It is due to the computation cost of solving the congruent system in PPMAC in signing process and generating primes in signing and verification process.

IV. Conclusion

This paper presented a new message authentication code that supported to verify the data integrity. This proposed mechanism preserves privacy of data against the verifier. Therefore, it is called privacy preserving MAC or PPMAC. The PPMAC was proved to be a CMA-secure MAC. Moreover, PPMAC only required partly information of message to verify the integrity instead of the whole original message. Such a PPMAC can be applied for a third-party auditor to audit data stored in cloud server so that the auditor learns nothing about the data [9].

References

[1] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997.

[2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*. Ieee, 2010, pp. 1–9.

[3] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology-ASIACRYPT 2008*. Springer, 2008, pp. 90–107.

[4] A. Giuseppe, B. Randal, C. Reza *et al.*, "Provable data possession at untrusted stores," *Proceedings of CCS*, vol. 10, pp. 598–609, 2007.

[5] K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409–428, 2012.

[6] G. J. Simmons, "Authentication theory/coding theory," in *Advances in Cryptology*. Springer, 1985, pp. 411–431.

[7] R. Crandall and C. B. Pomerance, *Prime numbers: a computational perspective*. Springer Science & Business Media, 2006, vol. 182.

[8] E. Bach, *Algorithmic Number Theory: Efficient Algorithms*. MIT press, 1996, vol. 1.

[9] H. V. Dang, T. S. Tran, D. T. Nguyen, T. V. Bui, and D. T. Nguyen, "Efficient privacy preserving data audit in cloud," in *Proceedings of 3rd International Conference on Computer Science, Applied Mathematics and Applications - ICCSAMA 2015*. Springer, 2015, pp. 185–196.