Journal of Korea Multimedia Society Vol. 11, No. 6, JUNE 2008(pp. 845-851)

Efficient OTP(One Time Password) Generation using AES-based MAC

Soon Dong Park[†], Joong Chae Na[††], Young-Hwan Kim[†††], Dong Kyue Kim[††††]

ABSTRACT

The ID/password method is the most classical method among authentication techniques on the internet, and is performed more easily and successfully than other methods. However, it is a vulnerable method against attacks such as eavesdropping or replay attack. To overcome this problem, OTP technique is used. The most popular OTP is HOTP algorithm, which is based on one-way hash function SHA-1. As recent researches show the weakness of the hash function, we need a new algorithm to replace HOTP. In this paper we propose a new OTP algorithm using the MAC(Message Authentication Code) based on AES. We also show that the new OTP outperforms HOTP experimentally.

Key words: OTP, HMAC, AES-CBC-MAC, AES-CMAC, AES-XCBC-MAC

1. INTRODUCTION

User authentication techniques on the internet can be classified into authentication by certificate and authentication by ID/password. Authentication by ID/password is the most classical method, which depends on the user's memory. The ID/password method is performed more easily and successfully than other methods, but it is vulnerable against attacks such as eavesdropping or replay attack. To overcome the weakness of ID/password method, OTP (One-Time Password) technique has been suggested and commercialized.

Recently HOTP(Hash-based One-Time Password) [1] was standardized by IETF. HOTP is an algorithm to generate one-time password based on HMAC-SHA-1 [2-4]. As, however, the weaknesses of the hash function are reported [5], we need a new algorithm to replace them.

In this paper we propose a new one-time password algorithm (called AES-OTP), based on AES(Advanced Encryption Standard) [6,7]. AES algorithm is a safe block encryption algorithm and has been accepted as a standard. Our algorithm utilizes primitives such as CBC-MAC [8], CMAC [9] and XCBC-MAC [10], which use AES instead of HMAC. We implement our new algorithm and the existing HOTP algorithm in both software and hardware, and compare their performances on a PC, various embedded environments and FPGA. Experimentally, AES-OTP shows better performance than HOTP.

This paper is organized as follows. We introduce HOTP and AES-based MAC in Section 2 and describe a new algorithm in Section 3. In Section 4, we give our experimental environments and results. Finally, we conclude with some remarks

※ Corresponding Author : Dong Kyue Kim, Address : (133-791) Division of Electronics and Computer Engineering Hanyang University, Seoul, Korea, TEL : +82-2-2220-2312, FAX : +82-2-2220-4926, E-mail : DQKIM@hanyang.ac.kr

Receipt date : Apr. 18, 2008, Approval date : June 4, 2008

[†] Division of Electronics and Computer Engineering, Hanyang University
(E-mail : sdpark@esslab.hanyang.ac.kr)
[††] Department of Computer Science and Engineering, Sejong University
(E-mail : jcna@sejong.ac.kr)
[†††] Cluem Co. Ltd.
(E-mail : younghwan.kim@cluem.com)
[††††] Division of Electronics and Computer Engineering, Hanyang University

※ This work was supported by "system IC2010" project of Korea Ministry of Commerce, Industry and Energy.

in Section 5.

2. PRELIMINARIES

In this section, we give brief introduction for HMAC and AES-based MAC.

2.1 HMAC(Hash Message Authentication Code)

HMAC [11] is a process of message authentication using cryptographic hash functions and a shared secret key between server and client. Cryptographic hash functions used as iterating a basic compression function, usually MD5 or SHA-1, that are applied on each block of data. Any iterative cryptographic hash functions may be used in HMAC, and HMAC can be classified by these different realizations, HMAC-MD5, HMAC-SHA1, and so on.

Let H be an iterative cryptographic hash function where data is hashed by iterating a basic compression function on blocks of data, and K be a secret key. We denote by B the byte-length of such blocks and by L the byte-length of hash output. B is 64 for most of the hash functions and L is defined by the characteristic of the selected hash function, e.g., L = 16 for MD5, and L = 20 for SHA-1. The secret key K can be of any length between L and B. If K is longer than B, we hash the key using H and use L-bit output as a new key value. Let K^+ be a B byte string created by padding an adequate number of zeros to the end of K. Then Figure 1 shows HMAC over the original data M.

$$\text{HMAC}_K(M) = H((K^+ \oplus opad)||H((K^+ \oplus ipad)||M));$$

where,
ipad = the byte 0x36 repeated B times
opad = the byte 0x5C repeated B times

Fig. 1. Progress of HMAC, where, ⊕ is bitwise XOR and || is concatenation.

2.2 AES

We describe AES block-encryption algorithm. Figure 2 shows AES encryption process. It generates initial round key stream, 9 repetition rounds and a final round. Each round consists of SubByte, ShiftRow, MixColumn and AddRoundKey transformations except the final round. The final round performs the same as the other round except the MixColumn transformation. Each operation is as follows:

i) SubByte operation substitutes independently and nonlinearly one byte on each state. The table for substitution is called S-box. In the finite field GF(2^8), multiplicative inverse is composed of mapping $x \rightarrow x^{-1}$ and affine transformation.

ii) ShiftRow operation exchanges the position of each byte in the state, without value modification.

iii) MixColumn transformation considers a row of state as polynomial on GF((2^8)^4) and performs the following formula, $b(x) = c(x) \otimes a(x)$.

iv) Finally, AddroundKey adds round key to all bytes of the states, and performs EXOR operation.

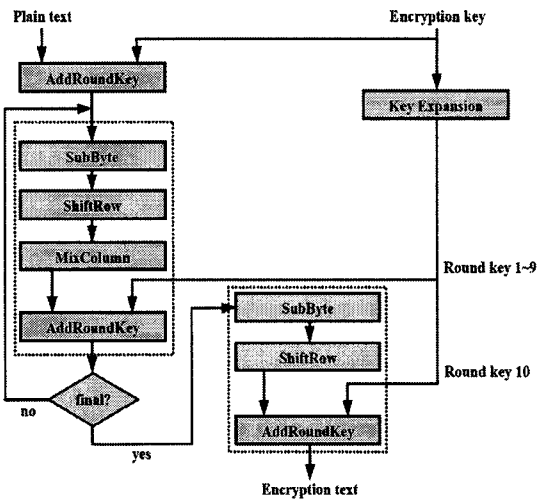

Fig. 2. AES encryption process

2.3. MAC based on AES

2.3.1 CBC-MAC

MAC is generated using a 128 bit key and message of random length. CBC-MAC generation process is shown in Figure 3. Input message is divided into n blocks, (M[1], ···, M[n]) of 128 bit length. If the length of M[n] is not 128 bits, pad {10i} to make it to 128 bits. We encrypt the message with the input key by CBC mode. The final block of encrypted message is used as MAC.

2.3.2 CMAC

CMAC consists of two parts, generating Sub-Key and MAC. Two Sub-Keys (*K1*, *K2*) are used in CMAC. *K1* and *K2* are used to operation when the last message is going to encrypt. Two Sub-Keys are defined in Figure 4. First, set R to $0^{120}10000111$ and L to $E_{key}(0^{128})$. If the most significant bit of L is 0, set *K1* to 1-bit left-shifted L value. If the most significant bit of L is 1, *K1*'s value is the result of exclusive-or operation of 1-bit left-shifted L value and R. If the most significant bit of *K1* is 0, *K2* is defined as 1-bit left-shifted *K1*. Otherwise, *K2* is defined as the result of exclusive-or operation of 1-bit left-shifted *K1* and R.

The stage of generating MAC has basically the same structure as that of CBC-MAC. Figure 5 shows the process of generating CMAC.

The details are as follows: Divide input message into n blocks of size 128 bits, and encrypt the first *(n-1)* blocks by CBC mode. If the size of the last block is 128 bits, encrypt the result of EXOR operation of *K1*, the n_{th} block, and the result of encrypting the first *(n-1)* blocks, and use it as MAC [Fig 5-(a)]. If the size of the n_{th} block is smaller than 128 bits, pad $\{10^i\}$ to make it to 128 bits, and then encrypt the result of EXOR operation block and *K2*, and use it as MAC [Fig 5-(b)].

2.3.3 XCBC-MAC

XCBC-MAC is also based on CBC-MAC. XCBC-MAC consists of two parts, generating Sub-Key and MAC. XBC-MAC uses three Sub-Keys (*K1*, *K2*, and *K3*). *K1* is used as a key in the process of block encryption, *K2* and *K3* are used in EXOR operation when encrypting final message. Three Sub-Keys are defined in Figure 6.

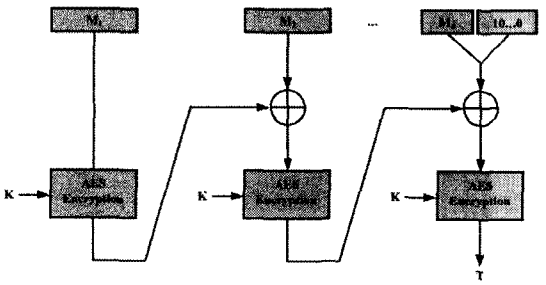

Fig. 3. Generating AES-CBC-MACK(M)

$R = 0^{120}10000111$
$L = E_{Key}(0^{128})$
If MSB1(L) = 0 then
    $K1 = L << 1$;
else
    $K1 = (L << 1)$ xor R
if MSB1($K1$) = 0 then
    $K2 = K1 << 1$;
else
    $K2 = (K1 << 1)$ xor R

Fig. 4. Sub-Key of XCBC-MAC

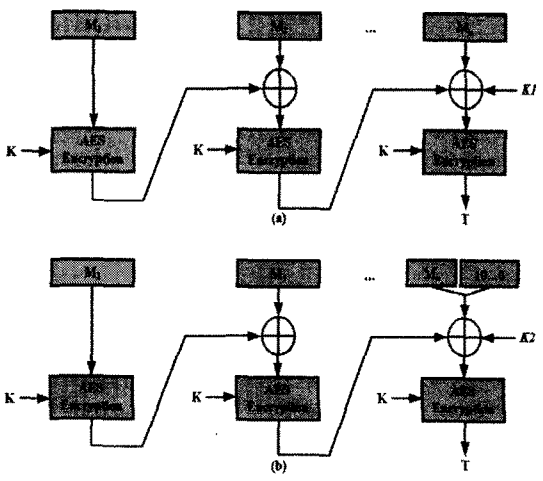

Fig. 5. Generating AES-CMACK(M)

$K1 = E_{Key}(ConstA)$
$K2 = E_{Key}(ConstB)$
$K3 = E_{Key}(ConstC)$

Fig. 6. Sub-Keys of XCBC-MAC

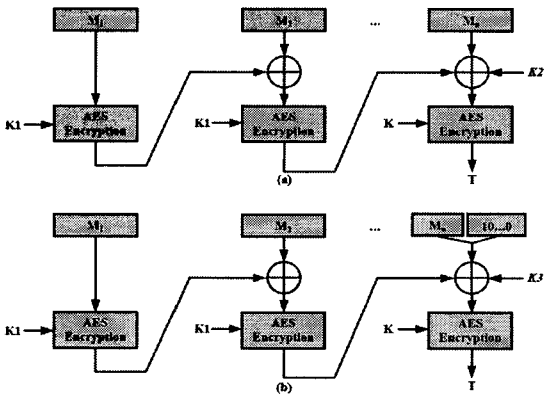

Fig. 7. Generating AES-CBC-MACK(M)

Generating MAC has the same structure as CBC-MAC. Figure 7 shows the development of XCBC-MAC. Developing process of XCBC-MAC is the same as that of CMAC, except using *K1* for AES's input key, *K2* and *K3* instead of CMAC's *K1* and *K2*, respectively.

3. OTP ALGORITHMS BASED ON HMAC AND AES

In this section we describe HOTP algorithm based on HMAC and AES-OTP algorithm based on various AES-based MACs.

3.1 Generating OTP based on HMAC

HOTP is the one-time password generating algorithm based on HMAC-SHA-1. Inputs are a key of random length and 64-bit counter. The output is 6~8-digit decimal numbers.

Input :
Key - a key, size of random length
Counter - a counter, size of 64-bit
Output :
HOTP - 6~8-digit decimal number

HOTP consists of three steps.

Step 1. *Generate HMAC value.*
HS = HMAC-SHA-1(Key, Counter)

Key is a shared secret key between client and server, and Counter is 8-byte counter value of the moving factor. This counter MUST be synchronized between client and server. Calculate HS value that is a 20-byte string.

Step 2. *Dynamic Truncation.*
int offset = HS[19] & 0x0F
DWORD P = HS[offset] ··· HS[offset+3]
SNum = P & 0x7FFFFFFF

Let offset be the low-order 4 bits of HS[19]. Generate a double word P from 4-byte of HS. Return the last 31 bits of P.

Step 3. *Calculate HOTP value.*
$HOTP = SNum \bmod 10^{Digit}$

HOTP is obtained as a congruent to SNum modulo 10^{Digit} and is an integer in the range of 0 and 10^{Digit}.

3.2 Generating OTP based on various AES-based MAC

The AES-OTP algorithms get a key and a counter as inputs, and produce an integer smaller than 10^{Digit}.

Input :
K - a key, same size to AES's input key.
C - a counter, size of 8 byte.
Output :
OTP - integer smaller than 10^{Digit}

The generation of OTP using AES-OTP algorithm consists of 3 steps.

Step 1. *Generate AES-MAC value*
HS := AES-MAC (K,C)

As described in the previous section, AES-

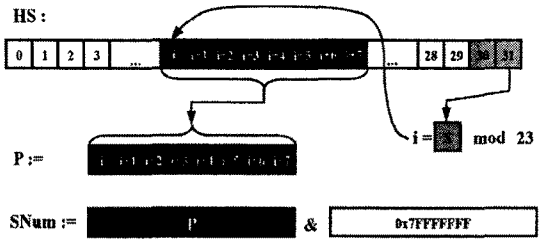

Fig. 8. Dynamic Truncation

CBC-MAC, AES-CMAC, and AES-XCBC-MAC can be used for AES-MAC. HS is a value of 16 bytes (128bits).

Step 2. *Dynamic Truncation*

Let i be the result on the 8 least significant bits of HS mod 23. Divide HS by 4 bits, and let P be the 8 blocks from the i_{th} block. SNum is defined as P of 31 bits, excluding the most significant bit of P. (See Figure 8.)

4. EXPERIMENTAL RESULT

We implement the proposed AES-OTP and existing HTOP in software, and analyze the performances. For AES-OTP, we implement and experiment three MAC generation algorithms, CBC-MAC, CMAC and XCBC-MAC. The experimental environments are as follows.

Environment 1 : PC
CPU - Intel Pentium IV 3.0 GHz
Compiler - Microsoft Visual Studio 6.0

Environment 2 : Embedded System
CPU - S3c2440A(ARM920T) 400 Mhz
Compiler - arm-linux-gcc 2.95.3

Environment 3 : FPGA
Language - VHDL
Target Device - Xilinx Virtex-II Pro xc2vp30 FF896-6

Environment 4 : Microprocessor
CPU -ATmega128 MCU(8Mhz) Atmel co.
Complier - C & Assembly Language

Table 1. Performance of HOTP and AES-OTP

(μsec)		Environment 1	Environment 2
HOTP		3.054	21.82
AES-OTP	CBC-MAC	1.089	5.34
	CMAC	1.449	8.90
	XCBC-MAC	1.973	11.44

Table 1 shows the experimental results on Environments 1 and 2. Both HOTP and AES-OTP take micro-sec time. AES-OTP shows about 1.5 ~ 4 times better performance than HTOP. This is caused by the operation time of CBC-MAC, CMAC and XCBC-MAC which takes much shorter time than HMAC. The differences in performance between the AES-OTPs are that CMAC needs to perform AES module 1 and 2 times more than XCBC-MAC and CBC-MAC, respectively.

Figure 9 shows the performances on throughput(Mbps) of various MAC modules implemented on FPGA. CBC-MAC, CMAC, and XCBC-MAC, are implemented based on AES using composite field GF($(2^4)^2$). The performance of AES-OTPs is much better than the OTPs based on MD5 and SHA-1. It also indicates that AES-based MACs show much better performance than hash based ones.

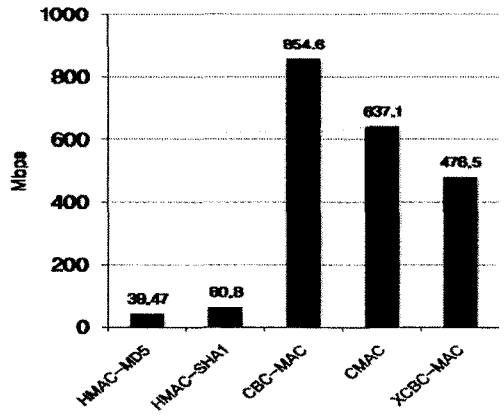

Fig. 9. Performance of various MAC modules using FPGA

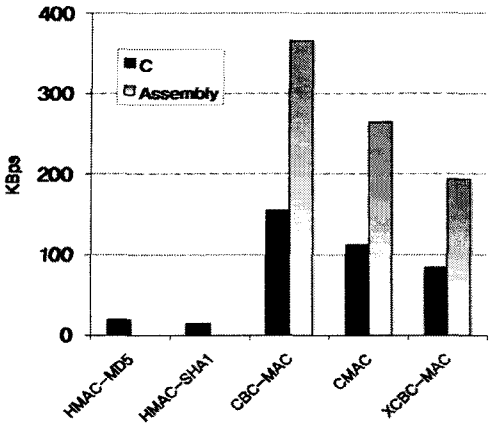

Fig. 10. Comparisons of throughputs.

Figure 10 shows the comparisons of throughputs between C language implementation and Assembly language implementation on the 8 bit processor, ATmega128. The implementation of Assembly language is a direct conversion of the C code into a hand-written assembly code. Processing speed of AES-OTP is better than that of HOTP. When implemented in Assembly, the throughputs are much faster and it is easier to adapt on 8 bit MCU.

5. CONCLUDING REMARKS

In this paper we proposed the OTP algorithm based on AES block-encryption algorithm instead of HMAC. According to the experimental results, generating AES-based OTP is superior to using HOTP algorithm. The security of the OTP algorithms using AES-based MAC depends on that of the underlying AES cipher. We leave the analysis of the safety of OTP using AES-based MAC as future works.

REFERENCES

[1] IETF RFC 4226, *HOTP: An HMAC-Based One-Time Password Algorithm*, December 2005.

[2] IETF RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*, February 1997.

[3] FIPS PUB 198, *The Keyed-Hash Message Authentication Code (HMAC)*, March 2002.

[4] FIPS PUB 180-2, *Secure Hash Standard*, Aug. 2002.

[5] Bruce Schneier, "SHA-1 broken," Feb. 2005, available at http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

[6] J.Daemen and V. Rijmen, "AES Proposal : Rijndael." *AES Proposal*, 1998.

[7] FIPS PUB 197, *Advanced Encryption Standard (AES)*, Nov. 2001.

[8] FIPS PUB 113, *Computer Data Authentication*, May 1985.

[9] FIPS PUB 800-38B, *Recommendation for Block Cipher Modes of Operation : The CMAC Mode for Authentication*, May 2005.

[10] IETF RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*, Sept. 2003.

[11] M.K. Lee, J.K. Min, S.H. Kang, S.H. Chung, H. Kim and, D.K. Kim "Efficient Implementation of Pseudorandom Function for Electronic Seal Protection Protocols," *LNCS*, Vol.4298, pp. 173-186, Springer, 2007.

Soon Dong Park

Soon Dong Park is a M.S. candidate in the Division of Electronics and Computer Engineering at Hanyang University, Korea. He received his B.S. degrees in Division of Electronics and Computer Engineering at Hanyang University in 2007. His research interests are in the areas of information security systems and ubiquitous sensor networks.

Joong Chae Na

Joong Chae Na received his B.S., M.S., and Ph.D. degrees from Seoul National University, Seoul, Korea, in 1998, 2000, and 2005, respectively. He is currently a full-time lecturer in Dept. of Computer Science and Engineering in Sejong University, Korea. His research interests include design and analysis of algorithms, and bioinformatics.

Young-Hwan Kim

Young-Hwan Kim received the B.S., M.S., in Public Administration from Hanyang University in 2000, 2004. and received PhD Candidate in Public Administration (e-Government and Regulation) from Hanyang University in 2006. From 2006, he was a General Executive Director at cluem co. ltd., Mobile ID Solution Provider. He is interested in the areas of cellphone security systems and ID systems.

Dong Kyue Kim

Dong Kyue Kim received the B.S., M.S., and Ph.D. degrees in dept. of computer engineering from Seoul National University in 1992, 1994, and 1999, respectively. From 1999 to 2005, he was an assistant professor in Division of Computer Science and Engineering at Pusan National University. He is currently an associate professor in Division of Electronics and Computer Engineering at Hanyang University, Korea. His research interests are in the areas of embedded security systems, crypto-coprocessors, and information security.