A Study on Efficient OTP Generation using Stream Cipher with Random Digit

Young Sil Lee*, HyoTaek Lim**, HoonJae Lee**

*Department of Ubiquitous and IT Graduate School of Design and IT, Dongseo University, Busan, 617-716, South Korea, Tel: +82-51-320-1730

**Div. of Computer and Information Engineering, Dongseo University

attract35@hotmail.com, htlim,hjlee@dongseo.ac.kr

Abstract— **The ID and password 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. It is a cornerstone of initiative for Open Authentication (OATH), it was published as an information IETF RFC 4226 in December 2005. As recent researches show the weakness of the hash function, to overcome this problem we need a new algorithm to replace them. In this paper we propose a method of creating one time password key using Ping Pong-128 stream cipher. Ping Pong is bit based stream cipher and it is designed with both security and efficiency in mind to satisfy the need for lightweight algorithm. We also use random digit, which it safe from attacks to creating a variable with the digit of the OTP.**

***Keywords*— OTP, HOTP, Stream Cipher, Ping-Pong**

I. INTRODUCTION

In recent years, the use of information and communication networks is being promoted in an increasingly diverse variety of ways, heralding a future in which networks will be everywhere. With the rapid development of computer network technologies, more and more networks connect together to exchange great information and share system resources. Security is an important issue for computer networks to prevent the information from being accessed by illegitimate or unauthorized users, remote authentication of users is certainly one of the most important services.

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 [1].

The elements of one time password system are a token included in a security/password algorithm or one time password key creating device, an authentication server and authentication client. One Time Password (OTP) is a password system where password can only be used once, and the user has to be authenticated with a new password key each time. One-time password can be generated by any kind of one-way hash function uses stream ciphers. To handle key management, the idea is to rely on a one-way hash function such as MD4, MD5 and SHA-1 algorithms. And the most popular OTP is HOTP algorithm, which is based on SHA-1. It is a cornerstone of initiative for Open Authentication (OATH), it was published as an information IETF RFC 4226 in December 2005.

The hashes are designed so it is very difficult to find two massages that produce the same hash, this is called "collision resistance". Because MD5 is 128-bit, by random chance you will find a collision by producing 2^{64} hashes. The weakness in MD5 is that a way has been found to produce such collisions with only 2^{42} hashes. This makes producing collisions practical. In order to overcome such weaknesses of the hash function, we need a new algorithm to replace them.

In this paper, we propose a method of creating one time password keys using Ping Pong-128 stream cipher. Ping Pong-128 is bit based stream cipher dedicated to hardware and to software. The Ping Pong-128 is designed with both security and efficiency in mind to satisfy the need for lightweight algorithm. And to further enhance the security of the OTP without having to set the digit between server and client, each session variable OTP to generate random digit approach offers.

This paper is organized as follows. We introduce OTP techniques, HMAC [2] and Ping Pong-128 algorithm [3] in Section 2. In Section 3, we describe HOTP [4-5], R(Random digit)-OTP [13] method and a new algorithm. And we conclude with some remarks in Section 4.

II. RELATED WORKS

In this section, we describe OTP techniques and synchronization techniques of OTP [6]-[10], HMAC [11] and Ping Pong algorithm [12].

A. OTP techniques

One Time Password (OTP) is a password system where passwords can only be used once, and the user has to be authenticated with a new password key each time. OTP has much stronger security because the user has to enter a newly created password key even if his key or password is exposed. The OTP is standardized by the IETF, and standardized again by verification related companies.

A lot of OTP solutions are secret and/or proprietary however some like OATH (usually HMAC-SHA1) are open-source and widely used and supported between OTP providers. OATH uses an event based OTP algorithm (it can also support time based however this isn't wide spread) which usually uses a secret character sequence (a.k.a. seed) only known to the two parties (being the user software/device and the OTP server) and the request number and sometimes other data such as customer unique seed, etc.. All this data is ran via the algorithm (usually HMAC-SHA1) to generate the OTP. RSA was one of the first companies to offer an enterprise OTP solution suing various software platforms and other form factors such as tokens. RSA uses a time based algorithm to calculate the OTP and as such requires the time to be synchronized between the client side and server side OTP components and has on the user side a time indicator so the user can time their password entry.

The major upside to time based OTP is than an event based token if someone got an OTP the only time limit they have to using it is until you generated a new OTP and used it. In a time based OTP one would only have a pre-defined time limit to use set OTP (e.g., 25 seconds) which would be more than enough at most instances. Event based OTP's on the other hand do not require time synchronization to take place and does not require the user to wait for a password to expire before entering it, etc...

B. Synchronization techniques

1) ***Challenge-Response type OTP:*** The OTP system generator passes the user's secret pass-phrase, along with a seed received from the server as part of the challenge, through multiple iterations of a secure hash function to produce a one-time password. After each successful authentication, the number of secure hash function iterations is reduced by one. Thus, a unique sequence of passwords is generated. The server verifies the one-time password received from the generator by computing the secure hash function once and comparing the result with the previously accepted one-time password.

2) ***Time-synchronized type OTP:*** The time-synchronized one-time passwords are usually related to physical hardware tokens (e.g., each user is given a personal token that generates a one-time password). Inside the token is an accurate clock that has been synchronized with the clock on the authentication server. On these OTP system, time is an important part of the password algorithm since the generation of new passwords is based on the current time rather than the previous password or a secret key.

3) ***Event-synchronized type OTP:*** Each time you ask an event based token for a new password, it increments the internal counter value by one. On the server, each time you successfully authenticate, the server also increments its counter value by one. In this way the token's and the server's counter values stay synchronized in lock step and always will generate the same one-time password. Event based tokens can get out of sync if the token is asked to generate a bunch of one-time passwords that are never actually used in authentication attempts. Then the token's counter value is increased while the server, oblivious, never increments his. Finally a token-generated one-time password is used for an authentication attempt but it fails because the server doesn't recognize it.

4) ***Time-Event synchronized type OTP:*** In both event-based tokens and time-based tokens it is possible for the server to auto-correct for synchronization problems, within certain limits. For event based tokens, the server always knows a lower bound on the current counter value (i.e., the counter value used in the previous successful authentication attempt) but not an upper bound. Therefore, if an unrecognized one-time password is seen, the server can try several counter values beyond its expected counter value to see if any of them match. If one happens to work, then the server knows that the intervening counter values have been "lost" and it should skip over them. For time based tokens, a similar strategy of trying out a few time intervals in the past and the future works to auto-synchronize with clock drift. Of course, regular use of the token is necessary to keep the drift within the recognized range.

C. HMAC(Hash Message Authentication Code)

HMAC is a process of message authentication using cryptographic hash functions and a shared secret key between server and client.

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 Table 1 shows HMAC over the original data M.

Table 1. Progress of HMAC, where, ⊕ is bit-wise XOR and || is concatenation.

$HMAC_k(M)$
$=H((K^+ \oplus opad)||H((K^+ \oplus ipad)||M))$;
where
ipad = the byte 0x36 repeated B times
opad=the byte 0x5C repeated B times

D. Ping Pong-128 Algorithm

Since the one time password mechanism is also a program, it is programmed to be random, but the randomness breaks after a certain period of time and passwords become predictable so having to exchange OTP modules after a certain period of time. To handle key management, the idea is to rely on a one-way hash function such as MD4, MD5 and SHA. Some weaknesses have recently been discovered in MD5 and SHA-1 algorithms.

Here we explain the Ping Pong-128 stream cipher and some analysis of generator. Stream ciphers are developed as approximation to behaviour one time code. The one-time password uses a long string of key stream which consists of bits that are chosen completely at random. This key stream is combined with the plaintext on a 'bit by bit' basis. The key stream is the same length as the message and can be used only once clearly a vast amount of key stream might be required. Ping Pong-128 is a specific cipher from the Ping Pong family of stream ciphers. It has two mutually clocking LFSTs and a single memory bit. The LFSRs are of lengths 127 bits and 129 bits. Ping Pong-128 has a two primitive polynomials $P_a(x)$ and $P_b(x)$ are following here:

$$P_a(x)=x^{127}\oplus x^{109}\oplus x^{91}\oplus x^{84}\oplus x^{73}\oplus x^{67}\oplus x^{66}\oplus x^{63}\oplus x^{56}\oplus x^{55}\oplus x^{48}\oplus x^{45}\oplus x^{42}\oplus x^{41}\oplus x^{37}\oplus x^{34}\oplus x^{30}\oplus x^{27}\oplus x^{23}\oplus x^{21}\oplus x^{20}\oplus x^{19}\oplus x^{16}\oplus x^{13}\oplus x^{12}\oplus x^{7}\oplus x^{6}\oplus x^{2}\oplus x\oplus 1$$

$$P_b(x)=x^{129}\oplus x^{125}\oplus x^{117}\oplus x^{113}\oplus x^{109}\oplus x^{105}\oplus x^{101}\oplus x^{97}\oplus x^{93}\oplus x^{89}\oplus x^{85}\oplus x^{81}\oplus x^{77}\oplus x^{73}\oplus x^{69}\oplus x^{65}\oplus x^{61}\oplus x^{57}\oplus x^{53}\oplus x^{49}\oplus x^{45}\oplus x^{37}\oplus x^{33}\oplus x^{29}\oplus x^{25}\oplus x^{21}\oplus x^{17}\oplus x^{13}\oplus x^{9}\oplus x^{5}\oplus 1$$

Together with the memory bit they give Ping Pong-128 an internal state of 257 bits. Ping Pong-128 takes a 128-bit key and a 128-bit initialization vector to fill the internal state [8].

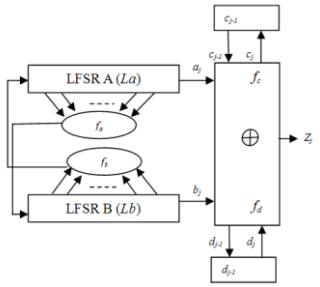

Figure 1. Ping Pong Family generator

The Ping Pong-128 OTP generator produces the output key stream by combining the LFSR sequences and the memory sequence. Ping Pong-128 has two mutually clocking LFSRs La and Lb, and a single bit of memory c.

$$f_a(L_a)=2L_{a42}(t)+L_{a85}(t)+1 \quad (3)$$
$$f_b(L_b)=2L_{b43}(t)+L_{b86}(t)+1 \quad (4)$$
$$c_j=f_c(a_j,b_j,c_{j-1})=a_j\,b_j\oplus(a_j\oplus b_j)c_{j-1} \quad (5)$$
$$d_j=f_d(a_j,b_j,d_{j-1})=b_j\oplus(a_j\oplus b_j)d_{j-1} \quad (6)$$

During this phase the whole key stream is generated constantly depending on the key and it's past.

The process is realized applying two similar functions named aj and bj, and defined as follows:

$$Z_j=f_z(a_j,b_j,c_{j-1},d_{j-1})=a_j\oplus b_j\oplus c_{j-1}\oplus d_{j-1} \quad (7)$$

Ping Pong-128 has a pair of LFSRs of different lengths that pair used a number of different feedback polynomials and took clocking taps from various stages of the registers.

As a result, It is therefore resistant against attacks based on basic key stream properties such as linear complexity and period. It defeats time-memory tradeoffs and clock controlled key stream generators.

III. A NEW OTP ALGORITHM BASED ON STREAM CIPHER WITH RANDOM DIGIT

In this section, we describe HOTP algorithm based on HMAC, R-OTP method and the proposed OTP algorithm using Ping Pong stream cipher with random digit.

A. HOTP(HMAC-based One-time Password)

The HOTP algorithm is based on an increasing counter value and a static symmetric key known only to the token and the validation service. In order to create the HOTP value, it will use the HMAC-SHA-1 algorithm, as defined in RFC 2104. Input is a key of random length and 64-bit counter. The output is 6~8-digit decimal numbers. HOTP consists of three steps.

Table 2. Input and output values.

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

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 mod 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}.

B. R(Random Digit)-OTP method

HOTP is the secure OTP generating used Hash-based MAC. However, HOTP has a fixed digit, it can be vulnerable for dictionary attack and guessing attacks. This method is safe from attack of another by creating a variable digit of the HOTP.

HOTP based on the existing OTP, it generates the HOTP and the same operation using SHA-1 algorithm.

1) *Assumptions*: R-OTP method is based on the following assumptions.

- Server and client share a unique secret key and the count is synchronized.
- The hash value HS must satisfy $HS>10^{Digit}$, HS of hexadecimal value to calculate the modular based decimal.

2) *ACT (Auto CounT):* R-OTP method eliminates the button of generates an event on the token, and count used automatically generated every 10 seconds in the time constant (usually 1 minute) to synchronize the value of event between server and the client.

3) *Random Digit:* The Digit which consists of OTP generation set the Q that do XOR between first and last numeral in the end 8 bit of the s bit. It should be satisfied $HS>10^{Digit}$ according to case of second assumption, therefore the Digit become 9 or under 9. So, if the case is R>9 and takes R mod 10 for generating Digit under 9.

C. Generating OTP based on improve Ping Pong with random digit.

The most popular OTP is used the one-way hash function. However as recent researches show the weakness of the hash function, we need a new algorithm to replace them. We used the PingPong-128 is one of a stream cipher and also used the random digit.

1) *Assumptions*: A proposed OTP algorithm is based on the following assumptions.

- Server and client share a Pin-ID of OTP generator and the count is synchronized.
- Created the PV value should be satisfied $PV>10^{Digit}$ and PV of hexadecimal value to calculate the modular based decimal.

2) *An improve OTP generation structure:* The KS() function is excluded can omit the part from existing PingPong-128 used in proposed algorithm. Key loading and part of the key renewal of stream cipher algorithm require many resources and operations. The propose of KS() function is to generated OTP, than encryption and decryption. Therefore, we omitted the key loading and part of the key renewal in the existing PingPong-128 algorithm and the two LFSR will be filled by input value as shown in Figure 2. Input is a user's ID and password, the OTP generate and the output is z_j by the operation with figure 2 and the same structure [14].

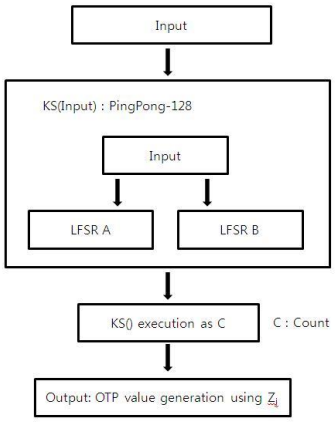

Figure 2. An improve OTP generation structure

3) *Synchronization method*: A proposed OTP algorithm used Time-Event method.

- Time count: The count used automatically generated every 30 seconds constant in the time (usually 1 minute) synchronized the value of the event between the server and the client.
- Event count: The count is generated when the event occurs.
  - When you receive the retransmission request constant in the time.
  - When the OTP input is normally performed or an error occurs in the input process.

4) *Generating OTP method*: A proposed OTP algorithm get a user's Pin-ID and a Time Stamp value as input, and produce an integer smaller than 10^{Digit}. The generation of OTP using an improve Ping Pong algorithm with random digit consists of 2 steps.

Step 1. Generate PV value.

PV = Ping Pong generator (Pin-ID||Time stamp)

The PV created using an improve Ping Pong stream cipher algorithm. The stream cipher has advantage that user can arbitrarily adjust the output. Therefore we omitted step 2, Dynamic Truncation for extracting the actual value of the used to generate of OTP in existing HOTP method, because Ping Pong generator can generate 32 bit used the OTP generated.

Input is user's Pin-ID and Time Stamp value. The output is key stream of z_j by the operation with figure 3 and the same structure. And the PV of 32 bit generated using z_j value.

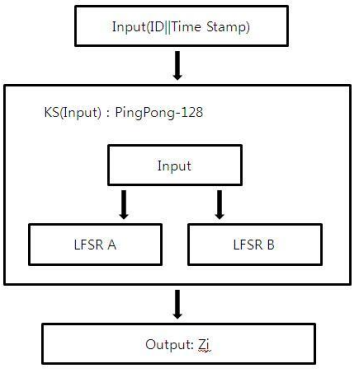

Figure 3. OTP generator

Step 2. Compute an OTP value

In this step, to generate OTP from calculated PV of 32 bit according to Digit, and the OTP is created PV mod 10^{Digit}.

The Digit which consists of OTP by R is created that do XOR between first and last numeral in the end 8 bit of the PV. It should be satisfied PV > 10^{Digit} according to case of second assumption, therefore the Digit 9 or under 9. So if the case is R > 9 and takes R mod 10 for generating Digit fewer than 9.

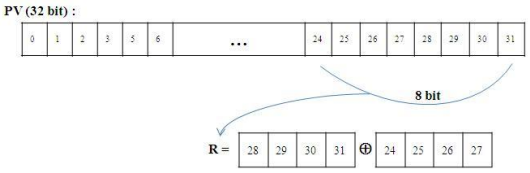

Figure 4. Compute an OTP value

IV. SECURITY AND EFFICIENCY ANALYSIS

In this section, we describe security and efficiency analysis of propose OTP generation algorithm.

A. Security Analysis

The key of the stream cipher is obtained using the function Ping Pong-128. For Ping Pong-128, both key and IV has a length of 128 bits, and together they fill 257 bit of internal state. The initialization process can also be used for rekeying. The process to generate the initial state for the key stream generator uses the generator itself twice.

By employing the Ping Pong algorithm itself, we take advantage of both the know security properties of the algorithm. And the design strength of Ping Pong-128 is 2128. It is therefore resistant against attacks based on basic key stream properties such as linear complexity and period.

Insecure online communication channels does not provide encryption, the value can be eavesdropping. But malicious attacker is difficult to dictionary attacks and guessing attack because variable digit in every session. It can't assume that the digit of OTP for use next session, hance the increases the consumed time and resources for attack use.

Also, the count is composed value of a variable by automatically update every 30 seconds. And if the used once after generate OTP, it will generate a new OTP raise to event. So we protected the replay attack.

B. Efficiency Analysis

The efficiency of existing operations to PingPong-128 stream cipher and the key renewal portion of the key loading was omitted, when the operation to process the number of bits to 4 bits were unified. And if case is Digit > 9 and takes mod 10 for generating Digit 9 or fewer than 9, because of the size is more than the value of PV.

Also, we omitted step 2, Dynamic Truncation for extracting the actual value of the used to generate of OTP in existing HOTP method by using stream cipher non SHA-1 hash function. And the performance of stream cipher is much better than the hash function. It also indicates that proposed OTP algorithm show much better performance than hash based ones.

V. CONCLUSIONS

One Time Password (OTP) is a password system where passwords can only be used once, and it can be generated by any kind of one-way hash function uses stream cipher. To handle key management, the idea is to rely on a one-way hash function such as MD4, MD5, and SHA. However, as recent researches show the weakness of the hash function. The hash function of one time password system can be seen as a module that can be replaced in case serious weaknesses are found in the hash function or when new more secure or efficient hash function are designed.

To overcome this problem, in this paper we proposed a new OTP algorithm using improved Ping Pong stream cipher algorithm and also used the random digit method.

Also it used the Time-Event method for synchronized between server and OTP generator, and it was complementing by created the OTP of a variable the digit to prepare for increasingly stronger attacks of attackers. Also, we increase to operation efficiency and can be reduced the overall performance of the process generation of OTP with an improved Ping Pong generator.

We leave the implementation of the proposed OTP algorithm using Ping Pong-128 stream cipher with random digit and authentication process between server and OTP generator as future works.

REFERENCES

[1] Soon Dong Park, Joong Chae Na, Young-Hwan Kim, dong Kyue Kim, "Efficient OTP(One Time Password) Generation using AES-based MAC," Journal of Korea Multimedia Society, vol. 11, No. 6, pp. 845-851, June. 2008.

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

[3] HoonJae Lee, Kevin Chen, "PingPong-128, A New Stream cipher for Ubiquitous Application," IEEE CS (ICCIT2007), ISBN: 0-7695-3038-9, pp. 1893-1889, Nov. 21023, 2007.

[4] IETF RFC 4226, HOTP: An HMAC-Based One-Time Password Algorithm, Dec. 2005.

[5] Bruce Schneier, "SHA-1 broken," Feb. 2005, Available : http://schneier.com/blog/archives/2005/02/sha1_broken.html
[6] IETF RFC 2289, A One-Time Password System, Feb. 1998.
[7] IETE RFC 1938, A One-Time Password System, May.1996.
[8] WIKIPEDIA, One-time password, Available : http://en.wikipedia.org/wiki/One-time_password#Time-sysnchronized
[9] (2009) mod-authn-otp, Available : http://code.google.com/p/mod-authn-otp/wiki/OneTimePasswords
[10] Erez Kalman, OTP , Available : http://sites.google.com/site/kalman/otp
[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.
[12] HoonJae Lee, Il Seok Ko (Franz Ko), "An Intelligent Security Agent for Reliable Cipher System using PingPong," Cybernetics and Systems Journal, vol. 39, No. 7, pp. 705-718, Oct.-Nov., 2008. ISSN 0196-9722.
[13] Soo-Young Kang, Im-Yeong Lee, "A Study on Secure R(Random digit)-OTP Scheme suing Random Digit," Proceedings of Korea Information Processing Society Conference, vol. 14, No. 2, pp. 1254-1257, Nov. 2007.
[14] Jang-Chun Lee, Hoon-Jae Lee, "OTP Authentication Protocol using PingPong-128," The Journal of the Korean Institute of Maritime Information and Communication Sciences, vol. 12, No. 4, pp. 661-669, 2008.