ISSN 2278-3091
Volume 8, No.4, July – August 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse35842019.pdf
https://doi.org/10.30534/ijatcse/2019/35842019

Security Evaluation of the Enhanced Key Generation Algorithm of Hashing Message Authentication Code

Enrique G. Abad[1], Dr. Ariel M. Sison[2], Dr. Ruji P. Medina[3]
[1]Technological Institute of the Philippines, Philippines, enrique.abad@dmmmsu-sluc.com
[2]Emilio Aguinaldo College, Philippines, ariel.sison@eac.edu.ph
[3] Technological Institute of the Philippines, Philippines, ruji.medina@tip.edu.ph

ABSTRACT

Quality assessment of the security of sensitive information being sent over or stored in a medium is vital in the world of computing and communications. Hash-based message authentication codes (HMACs) is a tool that provides such security check using a cryptographic hash function coupled with a secret key. In this paper, an enhanced key generation algorithm of HMAC-MD5 using XOR bitwise operator and left circular shifting is proposed. Six randomness tests from the NIST statistical test suite are performed to the proposed method which is implemented on the existing Travel Order Management System. Results showed that the binary sequence of the generated key is random in all tests as P-values passed the minimum significant level. The computed P-values are as follows: Frequency (Monobit) Test = 0.503, Frequency Test within a Block = 0.349, Run Test = 0.553, Test for the Longest Run of Ones in a Block = 0.022, Approximate Entropy Test = 0.325, and Cumulative Sums Test = 1.000. This means that the generator provides unpredictable random sequence of key.

Key words: circular left shifting, Hash-based message authentication codes, MD5, randomness, XOR bitwise operator.

1. INTRODUCTION

Cryptography is the most commonly used technique in protecting data from the hands of the attackers. The secrecy of the data is done through the translation of the information into unintelligibly form which could not be understood by anybody except the real sender and intended recipient which knows the secret key [1]-[5]. Among the many applications of this technique, authentication is widely used in various e-commerce activities, e-government business information, and internet information.

In cryptography, a Hash Message Authentication Code (HMAC) is a type of message authentication code (MAC) calculated using a specific algorithm involving a cryptographic hash function in combination with a secret key. As with any MAC, it may be utilized to validate both the authenticity and integrity of the message at the same time. Further, HMAC's cryptographic strength basically relies on the quality and size of the key, hash output length in bits, and the cryptographic strength of the chosen hash function [6], [7]. One of the well-known hash algorithms is MD5 due to its extensive availability and shorter length [8].

Since cryptanalysists are continuously drawing efforts in terms of designing different attacks to break hash algorithms, much work is required to increase the strength of the HMAC function so that its resistance will escalate against attacks. Much studies have proposed different techniques to make HMAC more resistant against attacks especially exhaustive and birthday attacks. To determine how secured these techniques are, various assessments or evaluations such as avalanche effect, brute-force attack, randomness, and cryptanalysis were performed to provide validation of their performance [9], [10].

Meanwhile, there are different thoughts, some think that in order to make HMAC non-vulnerable, existing underlying hash function in HMAC with more secure hash functions must be replaced, while others think that it is the key which should be random and unpredictable enough to decrease the vulnerability of the HMAC algorithm. As a matter of fact, random number is widely used in cryptographic applications, which is mainly used as key. In view of the fact that the security of the key totally depends on the amount and randomness of itself, it is very important to produce random numbers. Pseudo-random MD6 compression, large random prime numbers, and lookup tables using random numbers are the existing methods to generate a key which makes it more protected and harder to predict [11], [12].

Hence, this paper extends to measure and analyze the randomness test of our proposed key generation of HMAC-MD5 which is implemented to an existing application system [9].

2. EXISTING HMAC ALGORITHM

Naqvi and Akram (2011) proposed a modified HMAC to improve its strength against attacks (e.g. Birthday Attack and Exhaustive Key Search Attack) [11]. This served as the baseline of the modified HMAC algorithm. Its architecture is shown in Figure 1 and steps for computation of HMAC-MD5 are as follows:

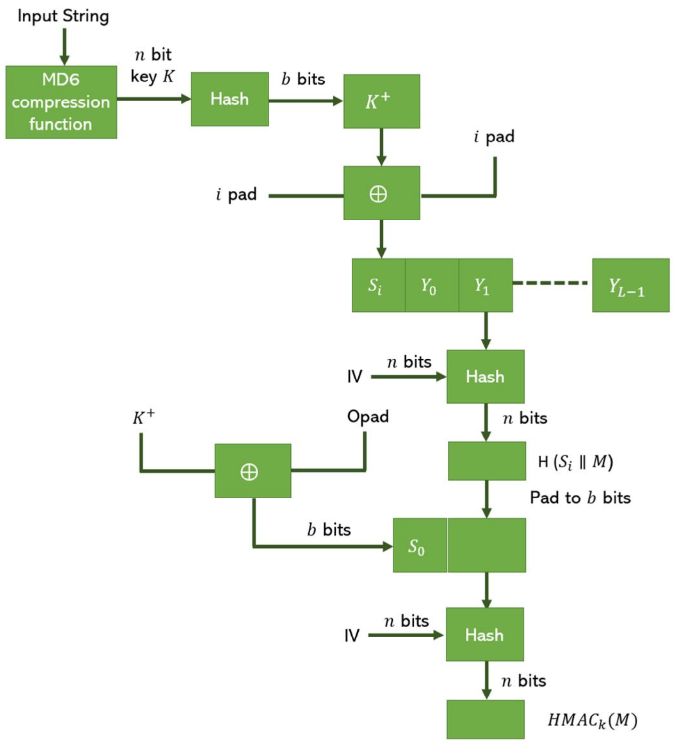

Figure 1: HMAC Algorithm [11]

a. Compute the secret key K by using MD6 function.
b. If the key has length longer that b (i.e. 512 bits), hash function h is used to hash the key K to a b bit long string K^+ or padded zeroes if the key is shorted than 512 bits.
c. XOR (bitwise exclusive-OR) K^+ with *ipad* value which is constant 0X36 repeated $\frac{b}{8}$ times to generate the block S_i of b-bit length.
d. Append m with S_i.
e. Operate h on the stream produced in step (c).
f. XOR K^+ with opad which is constant value OX5C to produce the block S_0 of length b-bit.
g. Attach the hash output computed in step (d) with S_0.
h. Apply h to the output computed in step (g) to finally get HMAC.

It has been proven that MD6 keeps some cryptographic properties such as pseudo randomness, unpredictability, parallelism, and others. Hence, the key generated by MD6 compression function becomes more random and harder to attack or forge. This implies escalating collision resistance of HMAC.

3. PROPOSED KEY GENERATION ALGORITHM

3.1 Framework

Figure 2 shows the flowchart of the proposed key generation algorithm to enhance HMAC. The key generation starts by hashing the plaintext using MD5 hash function. After getting the hash value of the plaintext, XOR bitwise operator was

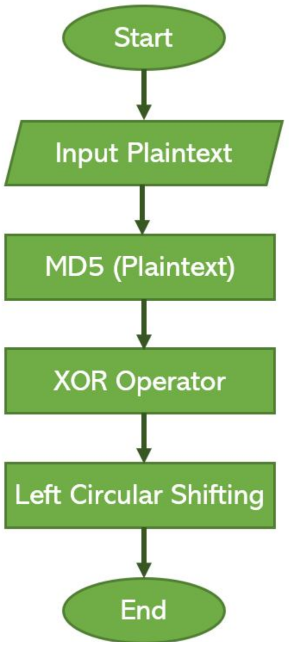

Figure 2: Flowchart of the Proposed Key Generation Algorithm of HMAC

applied between each character. Finally, the result underwent left circular shifting. Each major component of the proposed algorithm is briefly discussed below.

a. MD5

MD5 (Message-Digest algorithm 5) is commonly used cryptographic hash function whose main purpose is to examine the integrity of files. This hash function produces a 128-bit hash value, typically expressed as a 32-digit hexadecimal number, and has been used in a wide variety of security applications [13]-[15]. Users can always verify the integrity of the information after it is received since MD5 hash algorithm always generates similar output with that of given input. As a one-way operation, a hash function alters size of information into a shorter one with a fixed size to avoid similarity of characters [16]. Then, the resulting hash value is compared to the hash value being sent with information. If the two values are equal, this result entails that the information sent has not been modified. Therefore, the assurance of the integrity of the information is present.

MD5 cryptographic function operates 32-bit words. Let M be the message to be hashed. To make the length of the message (in bits) equal to 448 *mod* 512, the message is padded. The padded message must be 64 bits less than a multiple of 512. The padding is composed of single bit then pad enough zeros to attain the required length. Even if the length or size of M happens to be the same as 448 *mod* 512, padding is always utilized. The resulting bits should have at least one-bit of padding, and not more than 512 bits of padding. Then the length or size of the message (translated to bits) before padding is appended as a 64-bit block.

b. XOR Bitwise Operator

The following algorithm is used to perform XOR bitwise operator in generating the key of HMAC.

```
Input : hash value of plaintext n = h(plaintext)
temp = n[0]
for x = 1 to n.length -1

    temp= temp ⊕ n[x]
    v= v | temp

end for

temp = n[n.length-1] ⊕ n[0]
v=v | temp
```

It begins with accessing the first character of *n* and performing bitwise XOR (bit-level operation) to adjacent character. Each resulting value is concatenated to *v*. At the end, XOR bitwise operator is applied between the first and last character. Sample bitwise XOR operation is found in Table 1.

c. Left Circular Shifting

Left circular shifting is applied on each character of the XORed key. Afterwards, the number of shifts is calculated and convert each character into binary value. Finally, the resulting binary values are converted to corresponding ASCII characters. The following algorithm is utilized for left circular shifting operation. Figure 3 illustrates the graphical representation on how left circular shifting works.

```
Input : XORed key v=XOR(h(plaintext)),
        c_l_s=length(plaintext) mod 7
Output : Key generated
for x=1 to 32
    v[x]=v[x] << c_l_s
end for
k=v
```

Table 1: Sample Result of XORed Key

Key = 5D451402ABC4B2A76B9719D911017C592			
Value 1	**Operator**	**Value 2**	**Result**
5	XOR	D	q
q	XOR	4	E
E	XOR	1	t
…	…	…	…
…	…	…	…
f	XOR	5	S
XORed Key = qEt@pB└A┐6tF●0−D}J{B−?♬?☼> J␡Ft●			

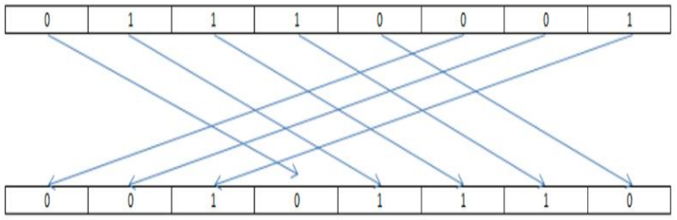

Figure 3: Process of Left Circular Shifting

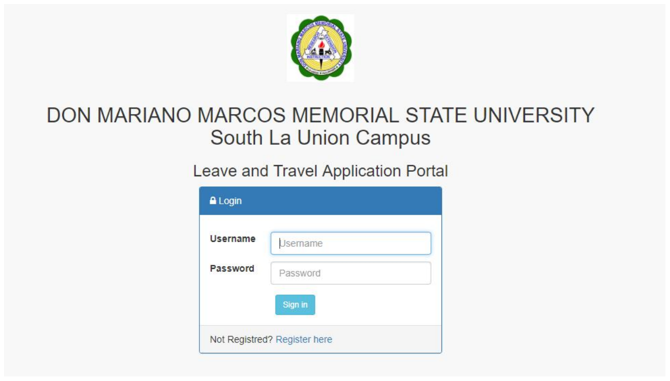
DON MARIANO MARCOS MEMORIAL STATE UNIVERSITY
South La Union Campus
Leave and Travel Application Portal

Figure 4: Login Page of the Travel Order Management System of DMMMSU-SLUC

After the design of the proposed key generation algorithm of HMAC, it was implemented into an existing Travel Order Management System (TOMS) of Don Mariano Marcos Memorial State University – South La Union Campus. Figure 4 shows the login page of the Travel Order Management System of DMMMSU-SLUC. In view of the fact that the proposed algorithm focused on password authentication, we gathered passwords of the faculty members and staff and digital signatures of the administrators of DMMMSU-SLUC. These data served as input to determine the performance evaluation of the proposed key generation algorithm. A total of 350 faculty members, 70 staffs, and 10 administrators have their own password to login to Travel Order Management System. These data were enough already to test the capability of the proposed algorithm against attacks.

3.2 Performance Evaluation

The proposed key generation algorithm of HMAC was evaluated through randomness tests which is also an important property of a cryptography algorithm block cipher to ensure that the algorithm is lack of pattern or predictability key. We adopted six tests among the 16 tests in NIST Statistical Test Suite [17] because of their applicability in the study.

a. Frequency (Monobit) Test

This test pertains to the proportion of zeros and ones for the entire binary sequence. Also, it determines whether the sum of zeros and ones in a binary sequence are approximately the same as would be expected for a truly random binary sequence. Frequency (Monobit) test assesses the closeness of the fraction of ones to $\frac{1}{2}$ wherein the count of ones and zeros in a binary sequence should be equal. All subsequent tests depend on the passing of this test; there is no evidence to indicate that the tested sequence is non-random. Equation 1 is used to calculate frequency test.

$$P - value = erfc\left(\frac{s_{obs}}{\sqrt{2}}\right) \tag{1}$$

where,

$$erfc = \frac{2}{\sqrt{p}}\oint_{z}^{\infty} e^{-u^2}du, s_{obs} = \frac{|S_n|}{\sqrt{n}}$$

b. Frequency Test Within a Block

This test is about the proportion of ones within M-bit blocks. It determines whether the frequency of ones in an M-bit block is approximately $\frac{M}{2}$, as would be expected under an assumption of randomness. To do this test, the following equation is used:

$$P - value = igamc(\frac{N}{2}, \frac{c^2(obs)}{2}) \tag{2}$$

c. Runs Test

This test corresponds to the total number of runs in the binary sequence, where a run is an uninterrupted sequence of identical bits. An exact *k* similar bit for a run of length *k* should be attained and is bounded before and after with a bit of the opposite value. Moreover, runs test aims to determine whether the expected number of runs of ones and zeros for a random binary sequence is the same as the computed number of runs for different lengths. Particularly, this randomness test confirms whether the oscillation between such ones and zeros is too slow or too fast. Equation 3 is applied to do the test.

$$P - value = erfc\left[\frac{|V_n(obs) - 2np(1-p)|}{2\sqrt{2np(1-p)}}\right] \tag{3}$$

d. Test for the Longest Run of Ones in a Block

The focus of the test is to check if the length of the longest run of ones within M-bit blocks is consistent with the expected length in a random sequence. It can be observed that an irregularity in the expected length of the longest run of ones would mean that there is also an irregularity in the expected length of the longest run of zeroes. For this reason, only a test for ones is required. Equation 4 is used to do this test.

$$P - value = igamc\left[\frac{K}{2}, \frac{c^2(obs)}{2}\right] \tag{4}$$

e. Approximate Entropy Test

Its main attention is the frequency of all possible overlapping m-bit patterns across the entire binary sequence. The purpose of the approximate entropy test is to compare the frequency of overlapping blocks of two consecutive or adjacent lengths $(m \, and \, m + 1)$ against the expected result for a random binary sequence. The following equation is used to complete approximate entropy test.

$$P - value = igamc(2^{m-1}, \frac{c^2}{2}) \tag{5}$$

f. Cumulative Sums Test

This test is related to the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted $(-1, +1)$ digits in the given binary sequence. It ascertains whether there is too small or too large cumulative sum of the partial sequences occurring in the tested binary sequence compared to the expected cumulative sum for random sequences. This cumulative sum may be allowed as a random walk. The excursions of the random walk should be near zero for a random sequence. On the other hand, the excursions of the random walk from zero will be large for certain types of non-random sequences. To do this test, (6) is applied.

$$P - value = 1 - \sum_{k=\frac{\left[\frac{n}{z}-1\right]}{4}}^{\frac{\left[\frac{n}{z}-1\right]}{4}} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right]$$

$$+ \sum_{k=\frac{\left[\frac{-n}{z}-3\right]}{4}}^{\frac{\left[\frac{n}{z}-1\right]}{4}} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right] \tag{6}$$

4. SIMULATION RESULTS

Considering the latter mentioned implementation scenarios, a thorough experimentation was carried out using the modified key generation of HMAC. Among the 16 tests in the NIST test suite, only six tests are appropriate in this study which are useful in studying and evaluating the binary sequences of the generated key. In table 2, results of these statistical tests are shown in which each P-value is the probability that a perfect key generator would have produced a binary sequence less random than the binary sequence that was tested, given the kind of non-randomness assessed by the test.

It is worthy to mention that all tests acquired a P-value less than 0.01. This means that the sequence of secret key is random. Surprisingly, the binary sequence of the key appears to have a perfect randomness for cumulative sums test as the computed P-value is equal to 1. Further, the results also show that the generator is suitable for cryptographic application in view of the fact that the generated keys are unpredictable.

Table 2: Results of Randomness Tests on the Proposed Key Generation of HMAC

Category	P-Value
Frequency (Monobit) Test	0.503
Frequency Test within a Block	0.349
Run Test	0.553
Test for the Longest Run of Ones in a Block	0.022
Approximate Entropy Test	0.325
Cumulative Sums Test	1.000

5. CONCLUSION AND FUTURE WORK

This paper proposed a modified key generation using HMAC-MD5 to increase the security of key against various attacks. The proposed method applied XOR bitwise and left circular shifting to produce a 255-bit hash size. To evaluate the binary sequences produced by the proposed key generator, six statistical tests were conducted. The results showed that the random number generator is secured enough for key generation which is implemented in the existing Travel Order Management System.

In future work, the optimization of the proposed key generation program is considered in hardware implementation. This might help in reducing memory and power consumption.

ACKNOWLEDGEMENT

The authors would like to thank Don Mariano Marcos Memorial State University - South La Union Campus for helping us to gather data. This work was supported by Technological Institute of the Philippines and Commission on Higher Education.

REFERENCES

1. D. J. S. Hombrebueno, M. G. C. E. Sicat, J. D. Niguidula, E. P. Chavez, and A. A. Hernandez. **Symmetric cryptosystem based on data encryption standard integrating HMAC and digital signature scheme implemented in multi-class messenger application**, in *Proc 2nd International Conference on Computer and Electrical Engineering*, Dubai, United Arab Emirates, 2019, pp. 327-334, IEEE.
2. J. A. Dev. **Usage of botnets for high speed md5 hash cracking**, in *Proc. 3rd International Conference on Innovative Computing Technology*, London, UK, 2013, pp. 314-320.
3. P. Amarendra Reddy and O. Ramesh. **Security mechanisms leveraged to overcome the effects of big data characteristics**, *International Journal of Advanced Trends in Computer Science and Engineering,* vol 8, no. 2, pp. 312-318, 2019.
4. A. E. Karrar and M. F. I. Fadl. **Security protocol for data transmission in cloud computing**, *International Journal of Advanced Trends in Computer Science and Engineering*, vol 7, no. 1, pp. 1-5, 2018. https://doi.org/10.30534/ijatcse/2018/01712018
5. R. S. Rubayya and R. Resmi. **Memory optimization of HMAC/SHA-2 encryption**, in *Proc. 2014 First International Conference on Computational Systems and Communications*, Trivandrum, India, 2014, pp. 282-287. https://doi.org/10.1109/COMPSC.2014.7032663
6. S. Idris, H. Zorkta, S. Khawatmi, and W. Aiyash. **Enhanced HMAC based upon 3-D rossler system**, in *Proc. International Conference on Future Computer and Communication*, Kuala Lumpar, Malaysia, 2009, pp. 465-469. https://doi.org/10.1109/ICFCC.2009.71
7. M. Najjar and F. Najjar. **d-HMAC dynamic HMAC function**, in *Proc. International Conference on Dependability of Computer Systems*, Szklarska Poreba, Poland, 2006, pp. 119-126. https://doi.org/10.1109/DEPCOS-RELCOMEX.2006.17
8. M. S. Vinola, A. M. Balamurugan, V. Chandana, and V. Durairaji. **Enhanced HMAC structure based burst header authentication design for optical burst switched networks**, in *Proc. 2nd IEEE International Conference on Recent Trends in Electronics Information and Communication Technology*, Bangalore, India, 2017, pp. 409-411. https://doi.org/10.1109/RTEICT.2017.8256628
9. E. G. Abad and A. M. Sison. **Enhanced key generation algorithm of hashing message authentication code**, in *Proc. 3rd International Conference on Cryptography, Security and Privacy*, Malaysia, 2019, pp. 44-48. https://doi.org/10.1145/3309074.3309098
10. X. Niu, Y. Wang, and D. Wu. **A method to generate random number for cryptographic application**, in *Proc. 10th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Kitakyushu, Japan, 2014, pp. 235-238. https://doi.org/10.1109/IIH-MSP.2014.65
11. S. I. Naqvi and A. Akram. **Pseudo-random key generation for secure HMAC-MD5**, in *Proc. IEEE 3rd International Conference on Communication Software and Networks*, Xi'an, China, 2011, pp. 573-577. https://doi.org/10.1109/ICCSN.2011.6014790
12. M. Bahadori, M. R. Mali, O. Sarbishei, M. Atarodi, and M. Sharifkhani. **A novel approach for secure and fast generation of RSA public and private keys on SmartCard**, in *Proc. 8th IEEE International NEWCAS Conference*, Montreal, QC, Canada, 2010, pp. 265-268. https://doi.org/10.1109/NEWCAS.2010.5603937
13. P. Ora and P. R. Pal. **Data security and integrity in cloud computing based on RSA partial homomorphic and MD5 cryptography**, in *Proc. International Conference on Computer, Communication and Control*, Indore, India, 2015, pp. 1-6. https://doi.org/10.1109/IC4.2015.7375655
14. H. Kumar, S. Kumar, R. Joseph, D. Kumar, S. K. S. Singh, and P. Kumar. **Rainbow table to crack password using MD5 hashing algorithm**, in **Proc. 2013 IEEE Conference on Information &**

Communication Technologies, Tamil Nadu, India, 2013, pp. 433-439. https://doi.org/10.1109/CICT.2013.6558135

15. X. Zheng and J. Jin. **Research for the application and safety of MD5 algorithm in password authentication**, in *Proc. 9th International Conference on Fuzzy Systems and Knowledge Discovery*, Sichuan, China, 2012, pp. 2216-2219. https://doi.org/10.1109/FSKD.2012.6234010
16. X. Nan-bin and H. Xiang-dan. **The mixed encryption algorithm based on MD5 and XOR transformation**, in *Proc. Second International Workshop on Education Technology and Computer Science*, Wuhan, China, 2010, pp. 394-396. https://doi.org/10.1109/ETCS.2010.127
17. L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. **A statistiacl test suite for random and pseudorandom number generators for cryptographic applications**, (No. Special Publication (NIST SP)-800-22 Rev 1a).