Increasing Cryptography Security using Hash-based Message Authentication Code

Seyyed Mehdi Mousavi*[1], Dr.Mohammad Hossein Shakour [2]

1-Department of Computer Engineering, Shiraz Branch, Islamic AzadUniversity, Shiraz, Iran .
Email : lord456@gmail.com

2-Assistant Professor, Department of Computer Engineering, Shiraz Branch, Islamic Azad University ,Shiraz ,Iran

Abstract

Nowadays, with the fast growth of information and communication technologies (ICTs) and the vulnerabilities threatening human societies, protecting and maintaining information is critical, and much attention should be paid to it. In the cryptography using hash-based message authentication code (HMAC), one can ensure the authenticity of a message.

Using a cryptography key and a hash function, HMAC creates the message authentication code and adds it to the end of the message supposed to be sent to the recipient. If the recipient of the message code is the same as message authentication code, the packet will be confirmed. The study introduced a complementary function called X-HMAC by examining HMAC structure. This function uses two cryptography keys derived from the dedicated cryptography key of each packet and the dedicated cryptography key of each packet derived from the main X-HMAC cryptography key. In two phases, it hashes message bits and HMAC using bit Swapp and rotation to left. The results show that X-HMAC function can be a strong barrier against data identification and HMAC against the attacker, so that it cannot attack it easily by identifying the blocks and using HMAC weakness.

Keywords: Cryptographic Security, Message Authentication Code, Expanded Hashing, HMAC, X-HMAC

1. Introduction

Nowadays, with the advancement of computers and the speed of sending data on the network, a lot of information is sent and received from a unit of time. Thus, given the necessity of information security in the network and, on the other hand, the necessity of speed in the exchange of information is one of the future needs of a two-way parallel cryptography algorithm for cryptography and decryption of bulk information. In the cryptography using HMAC, one can ensure the authenticity and reliability of a message. HMAC creates the message authentication code using a cryptography key and a hash function and adds it to the end of message to be sent to the recipient. If the message authentication code is identical to the message on the recipient's side, the packet is verified. Nowadays, HMAC is widely used in popular security protocols such as Ip-Sec, Transport Layer Security (TLS), and Secure Sockets Layer (SSL).

Given the conducted studies, some attacks have been made on HMAC in recent years, which could compromise the security of this widely used function [1, 2, 3, 4]. However, the attacks made were mostly due to the weakness in cryptography key used. In this function, the weakness is in the hashing function used in HMAC. In recent years, some studies have been conducted securing the key or using more secure hash functions, but unfortunately, it could not show the required efficiency and provide security [5, 6].

As stated, in the sent packet, the message and HMAC parts are recognizable in the packet. Using this feature, the attacker can make attacks like birthday attack and a quick attack or collision attack and can access the cryptography key. Moreover, by achieving cryptography key, manipulating the messages, and making HMAC using the available cryptography key, it sends the message towards the receiver, and the recipient does not notice the attack and confirms the message.

The advancement of cryptographyhas caused the emergence of different analysis methods, so that the encryption systems are intermittently broken. As all the experts in the field state that the security is never a hundred percent, but one can increase it in the network and other areas such as software by providing new security methods.

In this study, a complementary function called Extended HMAC (X-HMAC) is introduced by examining the structure, which uses two cryptography keys derived from the dedicated secret key of each packet and the dedicated cryptography key of each packet derived from the main X-HMAC key. In two phases of message bits and HMAC, it hashes by bit-turning and left-turning, which creates a strong barrier against the attacker in access to the well-formed structure and increases security of HMAC against these attacks.

2. Literature Review

Lee et al. (2017) conducted a study entitled “Analysis of coping with bit-flipping attacks on LoRaWAN Networks.” The paper tries to use hashing algorithm of block-data bits to repel bit-flipping attacks. According to the results of this study, one can understand that the above method can be used to improve the security of data transmission and to cope with bit-flipping attacks [7]. Joe et al. (2015) presented a study entitled “Analysis of first-order attacks on HMAC.” Given the identification of PIA and MAC blocks, the author has analyzed the possible attacks on HMAC [2]. Momeni and Taheri (2016) presented a study entitled “Time analysis attack on stream cipher algorithm.” In this paper, the vulnerability of a word-based stream cipher algorithm was examined from time-attack analysis point of view. By modifying this function and fixing the number of LFSR function clocks used, this attack could somehow be prevented [8].

Mahjan and Masih (2015) conducted a study entitled “Parallel Processing of BlowFish Algorithm.” In this study, the algorithm was explained and using CUDA, a part of this algorithm can be implemented in CPU and multiplied by theimplementation of speed [9]. Ayr et al. (2016) presented a hybrid model (symmetric and asymmetric) and could increase the data cryptography security using the proposed algorithm [10]. Thomas Perine and Lee Wong (2014) examined the likelihood of attacks like birthday attacks and brute-force. In both of them, attacks are doneas MAC bits and the message is specified in HMAC algorithm [11].

Dinor and Larent (2017) presented a study entitled “Improving public attacks against HMACs and HAIFA.” In this study, the main reason for the weakness of these two algorithms is mainly the hash function used; the shorter the hash function length, the hash function used has more security weakness [12]. Geo and Print (2014) stated two successful attacks against HMAC and NMAC, in which the second type of attacks even the length of the key used long ago, could not affect. Then by providing a new approach, they could prevent such attacks [13].

Jeong et al. (2013) showed that the recovery of HMAC / NMAC passwords is possible. Using their proposed algorithm, which slows down HMAC/NMAC process a bit, the authors of the paper could improve the cryptography key and enhance the security in the above algorithms [3]. Concini et al. (2006) considered HMAC and NMAC security research to use the hash function and collision functions hash SHA-0 and SHA-1, and MD4 and MD5 to recover keys in HMAC and NMAC [5].

Navi and Akram (2011) given the low security of MD5-SHA1 hash algorithms and vulnerability against attacks such as birthday attack and authors, present a solution to increase HMAC using random key generation using MD6 and its use in HMAC [6]. Satiov (2012) studied the Flame virus analysis [14].

Najjar et al. (2006) did a study entitled “Dynamic HMAC Function.” In this study, the researchers believe that using IV and dynamizing Outer pad (Opad) and Inner Pad (Ipad) can reduce attacks on HMAC, and the introduced algorithm can be a replacement for HMAC [15].

3. Proposed method

Given studies and surveys, the most significant cause of attacks to HMAC is the identification of the message packet and the HMAC. In the proposed method, we call X-HMAC, using a two-phase algorithm, whose first phase relates to bit Swapp of packet message, and the HMAC using RAM key, and in the second phase, the rotate packet is hashed using another cryptography key, can prevent the message packet and the Mac from detecting, and the attacker cannot detect and attack the message area and HMAC. At the end of the packet, the hashed message can be sent towards the receiver via one or more packets using the proposed algorithm. Using XHMAC cryptography key, the receiver is able to separate the message and the HMAC, and then, can examine the authenticity of the message using HMAC, and if the message is cryptography, after decoding the message after authentication, it decrypts the decryption key.

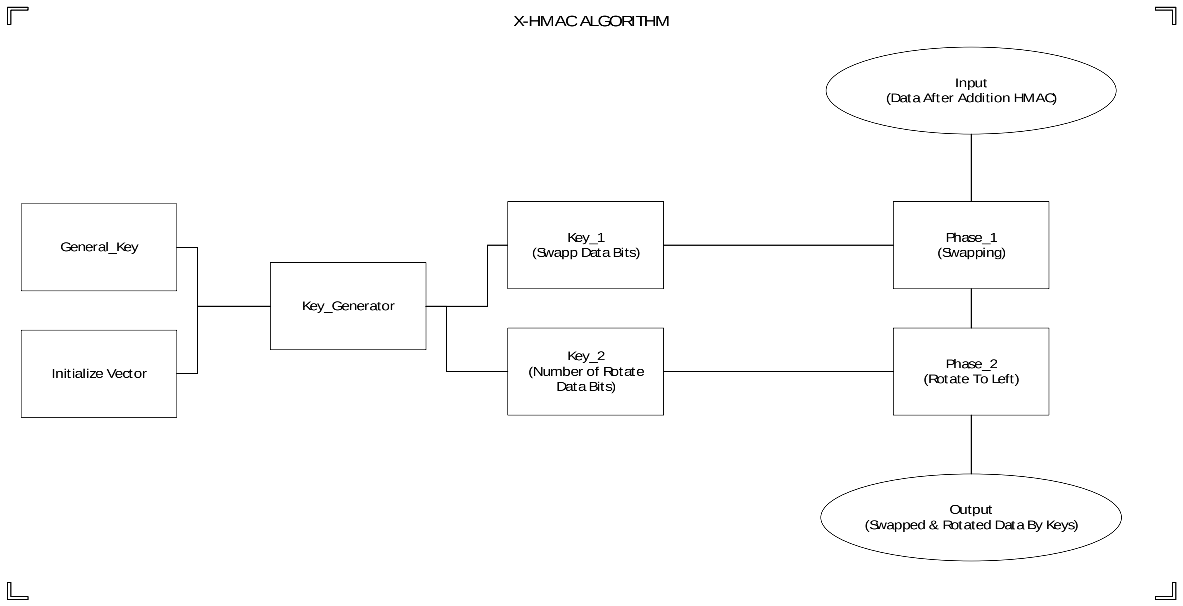


Figure 1: Extended HMAC pseudo-algorithm

1. First cryptography key

The first cryptography key algorithm is as follows:

```
        publicstring KeyGenarator(string IV, string XHMACSecret,int Xhmaclength)
                                                                           {
                                                  general GENERAL = newgeneral();
                var xhash = newHMACSHA512(Encoding.ASCII.GetBytes(XHMACSecret));
                  byte[] _xhmac = xhash.ComputeHash(Encoding.UTF8.GetBytes(IV));
                                                byte[] bar = newbyte[Xhmaclength];
                                   Array.Copy(_xhmac, 3, bar, 0, Xhmaclength);
                                                     return GENERAL.ToBinary(bar);
         }
```

The firstcryptography key is a key to the packet length (including HMAC and the message), using the HMAC-SHA256 approved by National Institute of Standards and Technology (NIST) organization.

In fact, this cryptography key determines which of the packet bits should be swapped. If

(1) xor (key (i), key (i + _rightbyte))

Equal to one, the swapping takes place between the bits in the first phase.

*rightbyte: specifies the middle of the key (in fact, the same is KeyLength / 2).

2. The secondcryptography key

This key is obtained using the firstcryptography key, and includes an integer, actually determines the number of turns to the left or right of the packet bits.

The algorithm to this key is as follows:

```
                                            publicint Key2(string key, int dataLength)
                          { returnConvert.ToInt32(key.Substring(Convert.ToInt32(key.Substring(
                                     dataLength / 3, 8), 2) % dataLength, 8), 2) % dataLength;
  }
```

The second cryptography key acts like this:

It calculates the correct value of the dataLength / 3 byte and obtains its residual to dataLength. Then, 8 bytes of the key is taken as that much and converted to integer; then, its remainder is calculated on the dataLength and the integer obtained determines the rotation to the right or left.

3. The first phase of cryptography

In the first phaseof cryptography, using the firstcryptography key explained above, it is determined whether i-bit and dataLength + I should be swapped or not.

The algorithm is as follows:

```
publicbyte[] Swap(byte[] data, string key)
{
int i;
int _dataLength = data.Length;
int _rightbyte = _dataLength / 2;
Parallel.For(0, i = (_rightbyte - 1), ctr =>//phase 1(swap phase)
{
if (xor(Convert.ToBoolean(Convert.ToInt16(key.Substring(i, 1))),
Convert.ToBoolean(Convert.ToInt16(key.Substring(i + _rightbyte, 1)))))
{
if (data[i] != data[i + _rightbyte])
SwapBytes(data[i], data[i + _rightbyte]);
}

});return data;
}
```

4. The second phase of cryptography

In this phase, rotation is done to the left equal to the number of secondcryptography key described in (secondcryptography key)the algorithm is as follows:

```
publicbyte[] RotateLeft(byte[] data, int key)  //phase 2:RotateLeft
{
int _datalength = data.Length;
byte[] data1 = newbyte[_datalength - key];
byte[] data2 = newbyte[key];
data1 = data.Skip(key).Take(_datalength - key).ToArray();
data2 = data.Take(key).ToArray();
Array.Copy(data1, data, data1.Length);
Array.Copy(data2, 0, data, data1.Length, data2.Length);
return data; }
```

5. The first decoding phase

As this phase is almost the same as the second phase of cryptography, it is possible to integrate the second phase cryptography function and the second decoding phase function and reduce the spatial complexity of X-HMAC algorithm. Here, for a greater clarity of this phase, we have prevented this.

The second phase algorithm is as follows:

```
        :RotateRight1publicbyte[] RotateRight(byte[] data, int key)  //phase
                                                                      {
                                               int _datalength = data.Length;
                                               byte[] data1 = newbyte[key-1];
                                   byte[] data2 = newbyte[_datalength - key + 1];
                             data1 = data.Skip(_datalength-key).Take(key).ToArray();
                                  data2 = data.Take(_datalength - key ).ToArray();
                                        Array.Copy(data1, data, data1.Length);
                             Array.Copy(data2, 0, data, data1.Length, data2.Length);
                                                                  return data;
                                                                      }
```

6. Second decoding phase

This phase is the same as the first cryptography phase and is decrypted using the first phase algorithm.

4. Results

It should be noted that the proposed two-phase X-HMAC algorithm is a step in increasing the security of the HMAC, and it alone cannot provide the cryptography security of information. In other words, X-HMAC algorithm is not a cryptography algorithm.

1. Examining the security of extended HMAC

No definitive security is possible in an algorithm, but solutions can be made to increase the security by identifying the problems and deficiencies of an algorithm. In this part, using the Cryptool software, we will implement a series of valid NIST tests and examine the results.

1.1. Birthday collision tests

For using 16-bit AES cryptography algorithm and HMAC-SHA256, 32-bit length, the length of the X-HMAC will be forty-eight bits (384 bits). The resistance and the hash needed for a successful attack to X-HMAC cryptography algorithm are calculated as shown in the following table.

Table 1: Birthdays collision

length of hex string	#bits	hash space size ($2^{\#bits}$)	Number of hashed elements such that (probability of at least one hash collision) = *p*									
			$p = 10^{-18}$	$p = 10^{-15}$	$p = 10^{-12}$	$p = 10^{-9}$	$p = 10^{-6}$	$p =$ 0.1%	$p =$ *1%*	$p =$ *25%*	$p =$ *50%*	$p =$ *75%*
8	32	4 × 3 × 109	2	2	2	2.9	93	2.9 × 103	9.3 × 103	5. 0 × 104	7.7 × 104	1. 1 × 105
16	64	1.8 × 1019	6. 1	1.9 × 102	6,1 × 103	1.9 × 105	6. 1 × 106	1.9 × 108	6. 1 × 108	3 .3 × 109	5. 1 × 109	7.2 × 109
32	128	3, 4 × 1038	2.6 × 1010	8.2 × 1011	2.6 × 1013	8.2 × 1014	2.6 × 1016	8.3 × 1017	2.6 × 1018	1. 4 × 1019	2. 2 × 1019	3. 1 × 1019
64	256	1.2 × 1077	4.8 × 1029	1.5 × 1031	4.8 × 1032	1.5 x 1034	4.8 × 1035	1.5 × 1037	4.8 × 1037	2.6 × 1038	4. 0 × 1038	5.7 × 1038
(96)	(384)	(3.9 x 10115)	8.9 × 1048	2.8 × 1050	8.9 × 1051	2.8 × 1053	8.9 x 1054	2.8 × 1056	8.9 x 1056	4.8 × 1057	7.4 × 1057	1. 0 × 1058
128	512	1. 3 × 10154	1.6 x 1068	2.5 × 1069	1.6 x 1071	5x2 × 1072	1.6 x 1074	5. 2 × 1075	1.6 x 1076	8.8 × 1076	1 × 4 × 1077	1.9 ×

In fact, we have calculated this resistance for the hash function used in HMAC, but given the use of key by HMAC, one can conclude that HMAC resistance to the birthday attack is greater than the hash function used.

2.1. Frequency analysis test

In this section, the result of one of the thousands of frequency analysis tests of a 128-bit string where initially encrypted by AES cryptography algorithm and then the authenticity of the message is added by HMAC by X-HMAC algorithm, is examined.

String before cryptography:

“aaaaaaaaaaaaaaa”

AES cryptography string (key: “1”):

“oklrUkBhTjItm5HFv2tAaw” ==

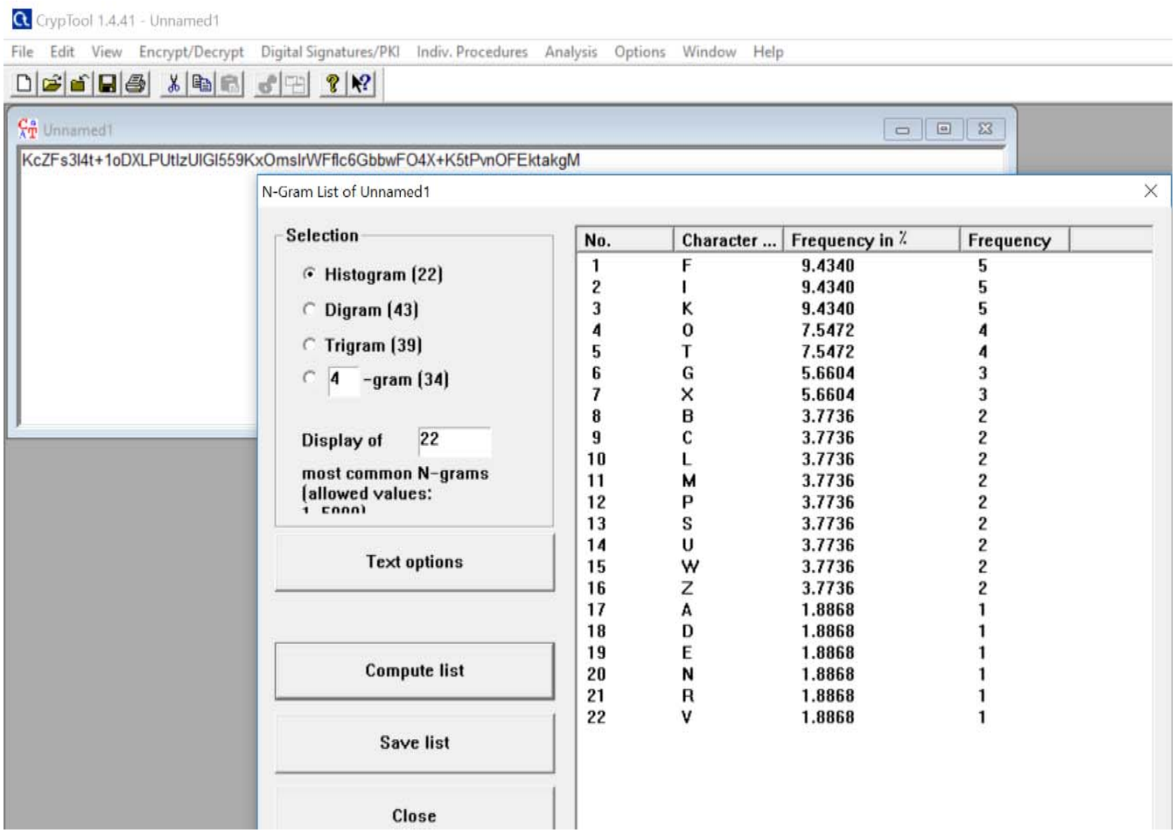
Strings obtained after applying HMAC (key: “1”):

“oklrUkBhTjItm5HFv2tAa5Z6lpGahA0vPPpWJ01hFawr5DnQzbeAp3C / xXNp985w”

Strings obtained after applying X-HMAC (key: “1”):

“KcZFs3I4t + 1oDXLPUtIzUIGl559KxOmsIrWFfIc6GbbwFO4X + K5tPvnOFEktakgM”

Frequency Analysis Test by Cryptool:



KcZFs3I4t+1oDXLPUtIzUIGl559KxOmsIrWFfIc6GbbwFO4X+K5tPvnOFEktakgM

Selection

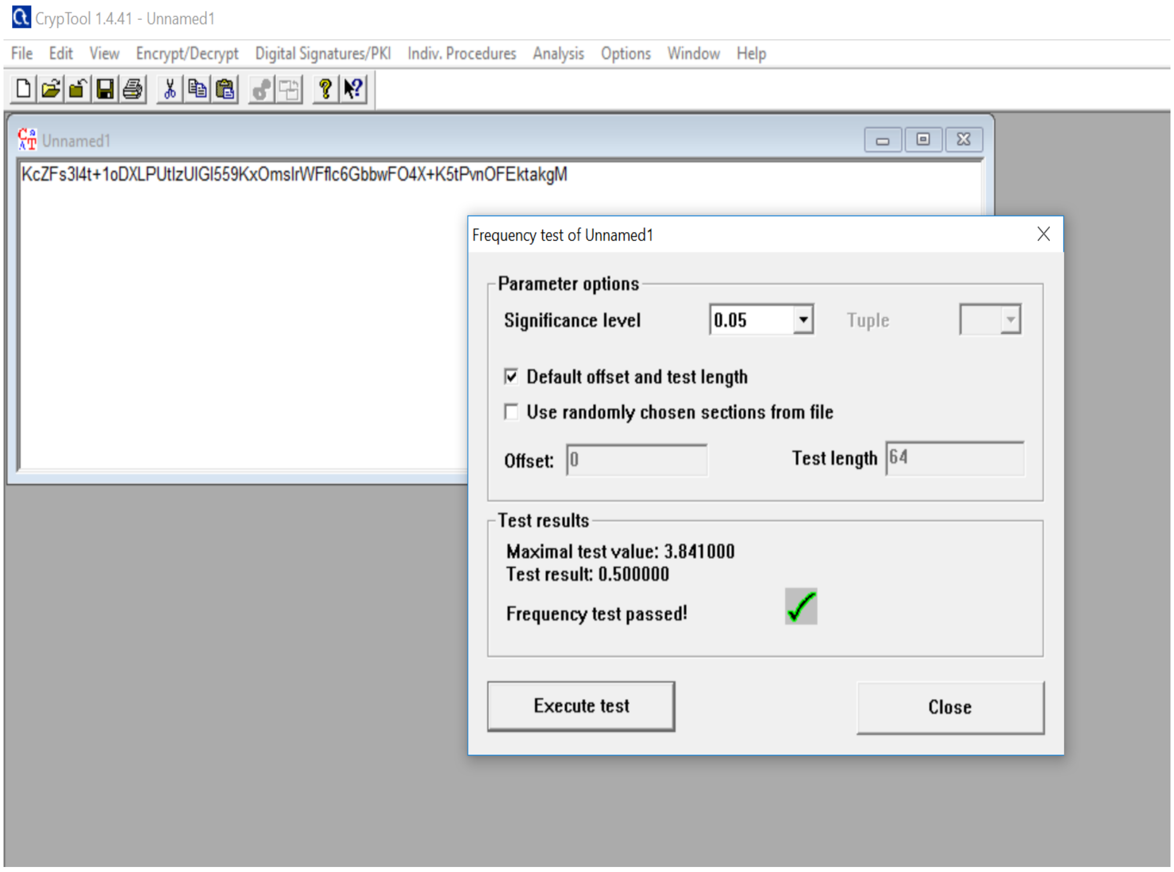
[✓ Histogram (22)] Digram (43) Trigram (39) 4 -gram (34)

Display of 22 most common N-grams (allowed values: 1..5000)

No.	Character ...	Frequency in %	Frequency
1	F	9.4340	5
2	I	9.4340	5
3	K	9.4340	5
4	O	7.5472	4
5	T	7.5472	4
6	G	5.6604	3
7	X	5.6604	3
8	B	3.7736	2
9	C	3.7736	2
10	L	3.7736	2
11	M	3.7736	2
12	P	3.7736	2
13	S	3.7736	2
14	U	3.7736	2
15	W	3.7736	2
16	Z	3.7736	2
17	A	1.8868	1
18	D	1.8868	1
19	E	1.8868	1
20	N	1.8868	1
21	R	1.8868	1
22	V	1.8868	1

Figure 2: An example of frequency analysis on extended HMAC

The result of the frequency analysis test by Cryptool:


CrypTool 1.4.41 - Unnamed1
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help
Unnamed1
Frequency test of Unnamed1
Execute test
Close

```
KcZFs3l4t+1oDXLPUtIzUIGI559KxOmsIrWFflc6GbbwFO4X+K5tPvnOFEktakgM
```

Parameter options
Significance level: 0.05
Tuple:
[✓] Default offset and test length
[ ] Use randomly chosen sections from file
Offset: 0
Test length: 64

Test results
Maximal test value: 3.841000
Test result: 0.500000
Frequency test passed! ✓

Figure 3: The result of an example of frequency analysis on extended HMAC

3.1. Test of data randomness

HMAC algorithm is safe against this test, and using Poker-Tests presents the test result of one of the thousands cases here.

String before cryptography:

“0123456789”

AES cryptography string (key: “1”):

“tIESavsn0Q0Jmpynfn80MA ==“

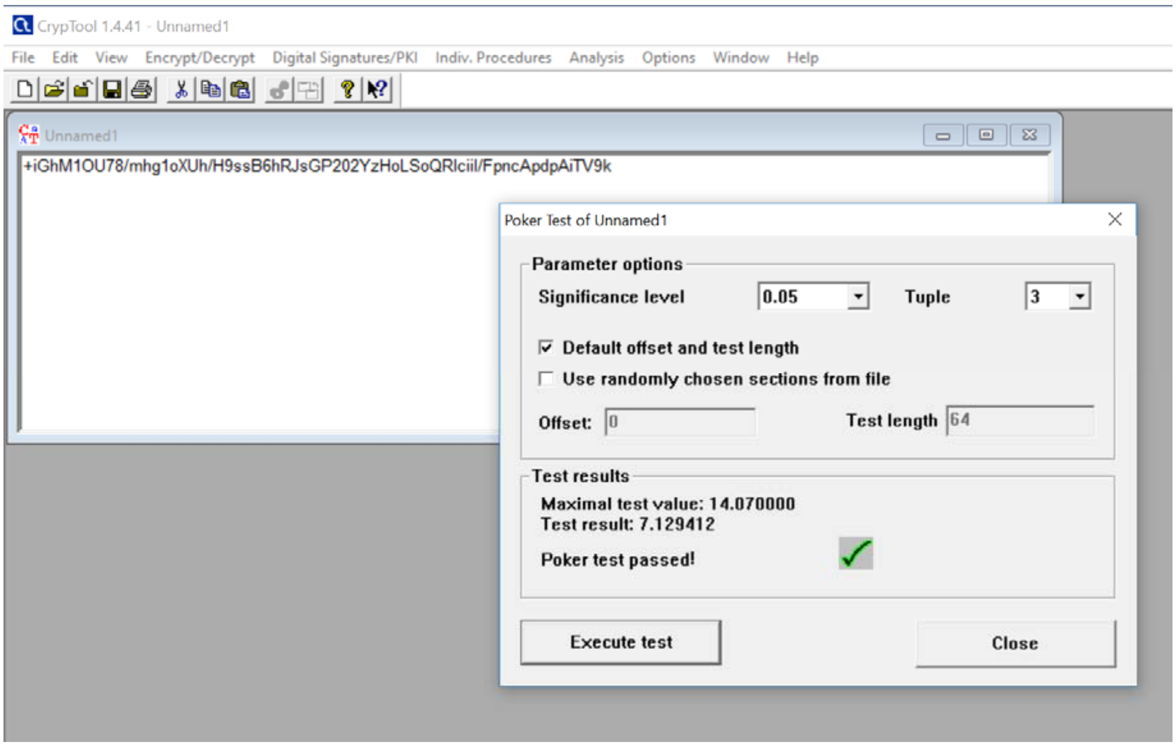
Strings obtained after applying HMAC (key: “1”):

“tIESavsn0Q0Jmpynfn80MGtC6kP4 / tlgPUKJNgx + 2mzGY9BaVCCJDkURH4tM7gFL”

Strings obtained after applying X-HMAC (key: “1”):

“+ iGhM1OU78 / mhg1oXUh / H9ssB6hRJsGP202YzHoLSoQRIciiI / FpncApdpAiTV9k”

The result of the poker test at 0.5 level is given in the figure below (this test is done by Cryptool).



+iGhM1OU78/mhg1oXUh/H9ssB6hRJsGP202YzHoLSoQRlciil/FpncApdpAiTV9k

Parameter options

Significance level 0.05 Tuple 3

☑ Default offset and test length

☐ Use randomly chosen sections from file

Offset: 0 Test length 64

Test results

Maximal test value: 14.070000
Test result: 7.129412

Poker test passed! ✓

Figure 4: The result of an example of poker test on extended HMAC

4.1. Collision test

In X-HMAC algorithm, as the output length depends on the length of HMAC and DATA, we will have:

$$Len_{X-HMAC} = Len_{Data} + Len_{HMAC} \ (2)$$

One can conclude that the proposed algorithm can increase the security against a collision attack, and this increase is more than HMAC security against collision, as HMAC-SHA256 algorithm is ultimately 256 bits long. However, using the same HMAC for example X-HMAC, we will have 128-bit data:

$$Len_{X-HMAC} = Len_{Data} + Len_{HMAC} = 128 \text{ b} + 256 \text{ b} = 384 \text{ b} \ (3)$$

5.1. Brute-force test

The number of brute-force attacks, if all output fields are exclusive to the hash function, it is calculated by the following formula:

$$(4) \text{ Number Of Brute\_Force\_Attack} = 2^n$$

In the previous form, *n is the number of output bits of a hash algorithm.

Number of search states to attack the hash algorithm HMAC-SHA512 (data length: 128 bits):

$$(5)\ 2^{512}$$

Number of search states to attack hash algorithm X-HMAC-SHA512 (data length: 128 bit):

$$(6)\quad 2^{512+128}$$

X-HMAC algorithm for a 128-bit string can increase security by 2^{640} times.

6.1. Test of avalanche effect

An avalanche test for a string is calculated using the Hamiltonian distance from the following equation.

Avalanche (i) =Hamming Distance (Hi ,Hj)(7)

Average Avalanche $= \frac{1}{n}\sum_{i=1}^{n}$ Avalanche (i) (8)

The results of the avalanche test are summarized in the following table with the implementation of C-Sharp environment.

Table 2: The result of a sample of an avalanche test on the extendedHMAC

Row	The range of implementation of test algorithm the d	X-HMAC key status	Input status	status IV	Test result of avalanche test(%)
1	0-1000	Variable (0-1000)	Fixed (input: 1)	Fixed (abcdefghijklm0001)	92.20
2	4000-5000	Variable (4000-5000)	Fixed (input: 1)	Fixed (abcdefghijklm0001)	91.12
3	3000-4000	Variable (3000-4000)	Fixed (input: 1)	Fixed (abcdefghijklm0001)	91.33
4	8000-9000	Variable (8000-9000)	Fixed (input: 1)	Fixed (abcdefghijklm0001)	93.04
5	0-1000	Random (88 bit)	Random (88 bit)	Random (string + number)	91.94

The higher the percentage of the avalanche test, the safer the algorithm will be. For instance, in the first row of this test, it is shown that on average for a 1000 range, with IV constant and input text, 92.20% of the input string has changed and this is very suitable for a cryptography algorithm. If this result is less than 80%, the algorithm is obsolete and unusable and over 90% shows an appropriate algorithm for the avalanche test.

2. Security of key cryptography

In our X-HMAC algorithm, we obtained a dedicated key with a length of 512 bits using the HMAC algorithm and the inclusion of the IV and the main X-HMAC key. The keys 1 and 2, used in the algorithm to determine Swapp and rotation of the bits, are derived from the dedicated key. It is obvious that if the keys derived or dedicated key of X-HMAC algorithm is disclosed using attacks like knowing the non-cryptographytext, the attacker cannot access the original key and cannot attack other data blocks through the dedicated key and break the X-HMAC of other blocks.

3. Comparison of the speed of using and not using the proposed algorithm

Using a system, we set up a constant string with a key and variable IV, during 1000 times of cryptography-computing implementation and adding hmac, and calculated the time of this process. Then, we computed the average run-time average of each run. Then we calculated the runtime of this process by adding x-hmac and the average of each run. As expected, the run-time due to the addition of the x-hmac function to the cryptography process has increased, but it is very low.

Table 3: Comparison of the speed of using and not using extended HMAC algorithm

The algorithmused	The running times	Total run time	Average approximate time of every run
HMAC	1000 times	59 seconds and 716 hundredths of a second	0.060 for each block
HMAC+X-HMAC	1000 times	66 seconds and 955 hundredths of a second	0.067 verses per block

4. Overall comparison of the use and non-use of the proposed algorithm

Here, according to some of the tests done on the experimental data input, we gather the result in the table below.

Table 4: Comparison of the use and non-use of extended HMAC algorithm

Methodology Name	HMAC	X-HMAC
Presentation year	1996	2018
Brief description	HMAC	HMAC and then applying the function in the proposed hash
Provider(s)	Bellare et al.	Mehdi Mousavi
Input length	Arbitrary (depending on hash algorithm)	Arbitrary (depending on hash algorithm)
Cryptography key length	Arbitrary (depending on hash algorithm)	Arbitrary (depending on hash algorithm)
Output length	Arbitrary 128-256-512 The HMAC algorithm is safe against attacks such as Pokers- Kasiski Frequency Analysis test	Equal to the length of the input
Resistance against the classical cryptography algorithm	✓ The HMAC algorithm is safe against attacks such as the Pokers- Kasiski Frequency Analysis test	✓ In addition to the security of the HMAC algorithm, it is used against attacks like the Pokers-Kasiski Frequency Analysis test, the security of the X-HMAC algorithm also increases the overall security of using the X-HMAC
Resistance against Frequency Analysis attack	✓ HMAC algorithm is safe against attack by frequency analysis.	✓ In addition to the security of the HMAC algorithm against the frequency analysis attack, the security of X-HMAC algorithm also increases the overall security of the X-HMAC.
Using Initialize Vector (IV) to prevent a key attack and revealing the HMAC key to all blocks	✗ Unfortunately, HMAC function does not allow the use of IV in the hmac function, and if an attacker accesses the key of one of the blocks, he will be able to attack all other blocks that use that key.	✓ By adding IV in the cryptography steps, if an X-HMAC key is released, a block is released, with the use of IV, the x-hmac key is reserved for other blocks, which increases the security of cryptography.
Hash resistance used against birthday collision attack (assumption: no collision in output hashes)	✓ The hash resistance of the hmac algorithm against the birthday collision attack is $2^{Len_{HMAC}/2}$	✓ In addition to maintaining the hash hysteresis of the hmac algorithm against birthday attack, this value has increased by the following x-hmac algorithm.
Hash resistance against brute-force attack (assuming no collision in output hashes)	✓ The hash resistance of the hmac algorithm will be against the brute-force attack: $2^{Len_{hmac}}$	✓ Using x-hmac, not only no failure occurs in hmac hash, but this resistance increases to the following value: $2^{Len_{hmac}+Len_{DATA}}$
Comparison of speed (This comparison is from the average run of 1000 rounds of cryptography then adding hmac)	0.060 seconds per data block	0.067 seconds per data block

5. Conclusion

The study dealt with using the studies done, the efficiency and widespread use of HMAC function in the famous protocols such as Ipsec, SSL, and using it to identify authenticity of the message. With the prevalence of using HMAC, the security of this algorithm is important and can greatly assist the cryptography community. It is proven that the security of the HMAC algorithm is greatly a function of the used hashing algorithm and its cryptography key, and the non-use of IV in this algorithm can affect the security of other blocks if the HMACcryptography key of one block is disclosed.

By studying previous studies to increase the length of the HMAC key to increase the security of this function, or to use safe hash functions or use of asymmetric algorithms instead of the abstract function, it has not been able to help HMAC security or reduce the critical speed and using asymmetric double-sided algorithms to encrypt bulky data is not cost-effective.

The study presented the message and HMAC bits prior to sending the proposed algorithm called X-HMAC. Using HMAC algorithm and IV and two dedicated keys derived from the original Brute-Force Attack key were hashed prior to sending (using the bit swapping and turning to the left of the bits) and then the data block was sent to the receiver.

If the attacker reaches the data block, he must first break and X-HMAC algorithm and decrypts it to obtain the correct layout of the block bit with the message and the HMAC, and then find a way to break HMAC. Clearly, the proposed X-HMAC algorithm can be a barrier against an attacker.

In the end, we state that the X-HMAC proposed function is written for bit hashing of HMAC in terms of the internal structure of the algorithm and cannot replace a cryptographic algorithm. However, its use as a complementary HMAC or using it in HMAC structure (to save memory and speed) can be suitable in improving HMAC security.

References

[1] Bellare, M., Canetti, R., & Krawczyk, H. (1996, August). Keying hash functions for message authentication. In Annual International Cryptology Conference (pp. 1-15). Springer, Berlin, Heidelberg.

[2] Guo, L., Wang, L., Li, Q., Zhang, Z., Liu, D., & Shan, W. (2015, March). A first-order differential power analysis attack on HMAC-SM3. In First International Conference on Information Science and Electronic Technology (ISET 2015). Atlantis Press.

[3] Jeong, K., Lee, Y., Sung, J., & Hong, S. (2013). Security analysis of HMAC/NMAC by using fault injection. Journal of Applied Mathematics, 2013.

[4] Kim, J., Biryukov, A., Preneel, B., & Hong, S. (2006, September). On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. In International Conference on Security and Cryptography for Networks (pp. 242-256). Springer, Berlin, Heidelberg.

[5] Contini, S., & Yin, Y. L. (2006, December). Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In International Conference on the Theory and Application of Cryptology and Information Security (pp. 37-53). Springer, Berlin, Heidelberg.

[6] Naqvi, S. I., & Akram, A. (2011, May). Pseudo-random key generation for secure HMAC-MD5. In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on (pp. 573-577). IEEE.

[7] Lee, J., Hwang, D., Park, J., & Kim, K. H. (2017, January). Risk analysis and countermeasure for bit-flipping attack in LoRaWAN. In Information Networking (ICOIN), 2017 International Conference on (pp. 549-551). IEEE.

[8] Momeni & Taheri (2016). Attack time analysis on a flow cipher algorithm. Electronic Defense and Cyber Defense Magazine, 4 (1).

[9] Mahajan, T., & Masih, S. (2015, September). Enhancing Blowfish file cryptography algorithm through parallel computing on GPU. In Computer, Communication and Control (IC4), 2015 International Conference on (pp. 1-4). IEEE

[10] Iyer, S. C., Sedamkar, R. R., & Gupta, S. (2016). A Novel Idea on Multimedia Cryptography Using Hybrid Crypto Approach. Procedia Computer Science, 79, 293-298.

[11] Peyrin, T., & Wang, L. (2014, May). Generic universal forgery attack on iterative hash-based MACs. In Annual International Conference on the Theory and Applications of Cryptographic Techniques (pp. 147-164). Springer, Berlin, Heidelberg.

[12] Dinur, I., & Leurent, G. (2017). Improved generic attacks against hash-based MACs and HAIFA. Algorithmica, 79(4), 1161-1195.

[13] Guo, J., Peyrin, T., Sasaki, Y., & Wang, L. (2014, August). Updates on generic attacks against HMAC and NMAC. In International Cryptology Conference (pp. 131-148). Springer, Berlin, Heidelberg.

[14] Sotirov, A. (2012). Analyzing the MD5 collision in Flame. Presentation at SummerCon, slides available at http://www. trailofbits. com/resources/flame-md5. pdf.

[15] Najjar, M., & Najjar, F. (2006, May). d-HMAC Dynamic HMAC function. In Dependability of Computer Systems, 2006. DepCos-RELCOMEX'06. International Conference on (pp. 119-126). IEEE.