Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp62-67)

Developing and Investigation of a New Technique Combining Message Authentication and Encryption

Eyas El-Qawasmeh and Saleem Masadeh
Computer Science Dept.
Jordan University for Science and Technology
P.O. Box 3030, Irbid 22110, Jordan

Abstrac:- This paper describes a new method for authenticating and encrypting messages. Our method employs any encryption algorithm as underlying block cipher. Proposed algorithm uses two key values, first key for the underlying encryption algorithm, and the second key for the new mode. The proven security describes the attacker inability to forge the new Encryption-Authentication algorithm, in terms of his (presumed) inability to break the underlying random S-box, the second key, and the underlying encryption function.

Keyword. Encryption, authentication, CBC, MAC.

1. Introduction

Encryption is used to insure privacy of data that is to possess data secret from public people other than its recipients. On the other hand, message authentication allows two participants sharing the key K to authenticate any transmissions between them. Message authentication is done by including a short string called "Message Authentication Code" (MAC) with every transmitted message.

The most dominant MAC is the "Cipher Block Chaining Message Authentication Code" (CBC MAC) which is stated in the International Standard ISO 9797 [1] and the U.S. Standard ANSI X9.9 [2]. In latest years, cryptographic hash has appeared, and became dominant.

The aim of the current work is to introduce a new technique, which has certain competence and safety measures advantages. New scheme is very simple that it is appear to believe one can with no trouble become aware of how to attack it. The success probability of the attacker in the new mode of operation is separate of the messages lengths. While the attacks of [3, 4] show that the success probability of the adversary in the CBC scheme increases linearly with the message length.

The organization of this paper will be as follows: section 2 is the proposed algorithm. Section 3 is performance analysis, and finally, section 4 is conclusions.

2. Proposed algorithm

Suppose a message M to be sent from one node of a network to another node. Proposed mode uses symmetric key setting, which represents a mapping from three-tuple input (message, key, S-box) to a binary decision; message authentic or not.

Following are some terms and concepts that are needed in presenting proposed algorithm:

- *Message length:* message length in bytes |M| should be greater than key length |K|.
- *Message formatting:* we assume the length of the message M is equal to |M| bytes at proposed encryption-authentication algorithm, and it should be a multiple of 8 bits without padding. A message is viewed as a sequence of 8-bit elements, M = M[1] M[2]…M [n].
- *Block size:* proposed algorithm is designed to work with variable block size. It is equal to the key length. For example, if the key K = 2549, then:

- Key length = 4 bytes
- Key digits are 2, 5, 4, and 9
- Digit size = 8-bits
- Block size=4Bytes= 8-bit*4=32 bits

In the proposed algorithm, the mode will be constructed from three components, which are KS, MAC, and VF. We will denote the MAC as MAC = (KS, MAC, VF) where:

- *Key scheduling (KS)*. Key scheduling algorithm schedules key value through proposed algorithm execution to generate new key for ciphering each data block.
- *MAC-generation (MAC).* MAC-generation inputs the message M ϵ {0, 1}*, and the key K as inputs to MAC function {MAC $\leftarrow$ $MAC_K(M)$}. MAC function will return a MAC ϵ {0, 1}*.
- *MAC-verification (VF).* MAC-verification inputs the shared key K, received message M ϵ {0, 1}*, and shared secret S-box as inputs to MAC-verification function {D $\leftarrow$ VF (K, M, MAC, S-box)}. Then, MAC-verification will return either one for acceptance, or zero for rejection.

The following section describes the sender role that will be carried on the sender side using proposed algorithm.

2.1 Sender procedure: When a sender needs to transmit his message M to a receiver, he must have all the requirements that enable him preparing his message M for sending. These requirements are:

1. Message M.
2. Shared symmetric key K, which is secret and stored in a file at each side.
3. Substitution box (S-box), which is secret and stored in a file at each side.

Proposed algorithm constructs S-box as 2-dimensional array that consists of 255 different values distributed over 15 columns and 17 rows according to one-byte variations. Therefore, each cell in the S-box contains a value between 1 and 255.

After retrieving any value, S-box values are shifted left one time from index 0 to the index of retrieved value. Changed S-box will be used at next encryption operation. Figure 2 shows an example of proposed S-box. The following is a description of proposed mode at sender side.

Following is a description of the proposed Encryption-Authentication algorithm.

Encryption-Authentication (Input: Original message, S-Box, Key**; Output:** MAC-File, Encrypted-file**)**

Begin

1) Read the secret key K, split it into digits, and stores each digit of it in key-array.
2) Order key-array digits in ascending order, and store them in key-Order array.
3) While (! Message.eof ())

{

For counter = 1 to key-length

{

a) Read one byte of the message.
b) XOR read byte with the corresponding key digit at the key-array.
c) Store the result of the XOR operation in XOR-array.

} ***// End of "For counter = 1 to key-length" loop***

Order XOR-array based on corresponding key-order array, and store it in ordered-array
// MAC generation
MAC generation:
For each stored value at ordered-array, compute the following:

- Number of its repetitions from the beginning of the message.
- Compute the following:

*P-MAC = Ordered[i] * Ordered [i+1] mod 255 + Number of repetition*

- *MAC = MAC + P-MAC*

After finishing all ordered-array values, message tag will be the value of MAC

// Ciphertext generation

Encrypted message generation: use each stored value at ordered-array as an index to retrieve the value stored in the corresponding S-box index as follow:

For each ordered-array[i] value, where i = 1, 2,…,|M|

- Encrypted-value = S-box [ordered-array [i]]
- Write the value of "Encrypted-value" on Encrypted file.
- Rotate stored S-box table values from index 0 to the index of the current ordered-array[i] value.

Start key scheduling

For I = 1 to key-length -1

{Key-array [i] = (Key-array [i] XOR Key-array [i+1]) + S-box [Key-array[i]]}

Key-array [|K|] = Key-array [0] XOR Key-array [|K|] + S-box [Key-array[i]]

} ***// End "while (Message.eof)" loop***

4) Key replacement: use the last scheduled key digits as indices to S-box table to replace the current key-array values as follow.

For d = 1 to key-length

Key-array[d] = S-box [Key-array[d]]

New key digits will be used for encrypting and authenticating next message using the proposed algorithm.

5) Send original message with the corresponding MAC value to the receiver.

END // End of encryption-authentication process

Example: Suppose the sender wants to send a message M = "*Standard*" to the receiver. The sender will use a randomly key K =*"14Sd7rgw"* and random S-box that will be shown at figure 2:

234	106	200	162	93	138	148	13	203	134	105	232	130	116	19
12	75	221	141	118	253	230	157	136	182	77	37	108	96	20
29	227	85	60	109	35	226	1	220	8	212	165	196	100	186
68	242	66	64	150	187	176	252	72	45	222	71	195	26	49
61	51	97	70	159	191	207	167	44	89	217	205	171	122	248
17	90	33	14	181	56	114	209	149	161	121	194	190	42	192
155	202	163	128	154	241	126	4	119	244	110	58	83	125	185
30	137	144	102	78	231	46	139	63	143	184	22	215	81	91
69	55	80	5	67	9	214	129	18	120	246	151	198	208	21
32	73	3	183	39	140	28	50	92	31	112	76	174	10	101
201	180	247	235	107	147	74	41	224	206	82	245	98	34	158
104	15	175	86	251	36	43	233	249	164	24	123	146	11	79
179	103	38	23	170	7	25	124	160	216	142	193	238	189	255
153	62	223	88	135	53	95	254	199	219	48	239	59	173	236
228	52	168	40	87	204	240	117	243	6	229	57	132	84	169
210	127	250	113	218	2	211	115	145	94	225	111	188	197	47
131	27	237	16	54	178	99	166	172	177	156	213	152	65	133

Figure 1: Random generated S-box

The following explanation shows how the sender will encrypt and authenticate the message M.

Message M = *"Standard"*	**The key K =** *"14Sd7rgw"*
First message block: *"Standard"*	**Key elements:** *1 4 S d 7 r g w*
First block in ASCII representation: *83 116 97 110 100 97 114 100*	**Key elements in ASCII representation:** *49 52 83 100 55 114 103 119*
XOR-ing plaintext bytes with the corresponding key elements: *83 XOR 49 = 98* *116 XOR 52 = 64* *97 XOR 83 = 50* *110 XOR 100 = 10* *100 XOR 55 = 83* *97 XOR 114 = 19* *114 XOR 103* = 21 *100 XOR 119* = 19	
XOR-array contents: *98 64 50 10 83 19 21 19*	**Key-Order:** *49 52 55 83 100 103 114 119*
Ordered array: *98 64 83 50 10 21 19 19*	
Generated ciphertext bytes: *Encrypted = S-box[98] = 119* *Encrypted = S-box[64] = 191* *Encrypted = S-box[83] = 161* *Encrypted = S-box[50] = 72* *Encrypted = S-box[10] = 19* *Encrypted = S-box[21] = 77* *Encrypted = S-box[19] = 182* *Encrypted = S-box[19] = 148*	
MAC computation: *Compute = 98 * 64 mod 255 + 1 = 153* *MAC = 0 + 153 = 153* *Compute = 64 * 83 mod 255 + 1 = 213* *MAC = 153 + 213= 366*	

*Compute = 83 * 50 mod 255 + 1 = 71 MAC = 366 + 71 = 437* *Compute = 50 * 10 mod 255 + 1 = 246 MAC = 437 + 246 = 683* *Compute = 10 * 21 mod 255 + 1 =211 MAC = 683 + 211 = 894* *Compute = 21 * 19 mod 255 + 1 = 145 MAC = 894 + 145 = 1039* *Compute = 19 * 19 mod 255 + 1 = 107 MAC = 1039 + 107 = 1146* *Compute = 19 * 98 mod 255 + 2 = 79 MAC = 1146 + 79 =* **1225** **Thus 1225 will be sent to the receiver as generated MAC value.**
Scheduled key: *key[0] = (49 XOR 52 + S-box[49]) mod 255= 77* *key[1] = (52 XOR 83 + S-box[52]) mod 255= 70* *key[2] = (83 XOR 100 + S-box[83]) mod 255= 0* *key[3] = (100 XOR 55 + S-box[100]) mod 255= 193* *key[4] = (55 XOR 114 + S-box[55]) mod 255= 95* *key[5] = (114 XOR 103 + S-box[114]) mod 255= 164* *key[6] = (103 XOR 119 + S-box[103]) mod 255= 141* *key[7] = (119 XOR 49 + S-box[119]) mod 255= 161*
Key replacement: *Key-array[0]=S-box[77]=181 Key-array[1]= S-box[70] = 171 Key-array[2] = S-box[0]=203* *Key-array[3]= S-box[193]=189 Key-array[4]= S-box[95]=126 Key-array[5]=S-box[164] =158* *Key-array[6] = S-box[141] =28 Key-array[7]= S-box[161]=245* **New key that will be used with next message M is:** *181 171 203 189 126 158 28 245*

The change that has happened to S-box indices and to key elements presents a fuzzy situation. The key elements that were used for encrypting the first plaintext block are not used for encrypting the second plaintext block. Key scheduling removes any pattern in the message that is to be encrypted.

2.2 Receiver procedure

In case of sending plaintext message, the procedure followed by the sender will be followed by receiver with comparing received and computed MAC value. Depending on application, receiver may have to decrypt the received message. Therefore, the receiver procedure will be as follow:

Verify (Input: Message, MAC; **Output:** "Verifiable" or "Not Verifiable"**)**

Begin

For I = 0 to |M|

1. Search S-box values that are found at the receiver side to get the index of 8-bit of the received ciphertext block.
2. Rotate S-box values from index 0 to the retrieved index.
3. Reorder retrieved indices corresponding to the key digits order.
4. XOR each key digit with the corresponding retrieved index value message.

Next I

End

3. Performance analysis

A comparison between CBC-MAC (with DES and Triple DES (TDES) as underlying encryption algorithms) and the proposed algorithm that focuses on MAC value generation will be presented. The points that we will investigate are:

3.1 MAC generation, (before or after encryption)

There were deep deliberations whether it is better to MAC the plaintext or the ciphertext. Krawczyk and Bellare Namprempre suggested applying encryption, followed by authenticating the ciphertext, which is called "Encrypt Then Authenticate (ETA)" [8, 9]; it was secure all the time. However, authentication before encryption of the plaintext might not be secure, even if the encryption and authentication algorithms were independently secure [8, 9]. In the CBC-DES, CBC-TDES, and the proposed algorithm, the use of encryption followed by message

integrity function will keep the previous mentioned advantages.

3.2 Execution time experiment

The first experiment will be execution time measure of CBC-DES, CBC-TDES, and the proposed algorithm to generate MAC value and ciphertext. CBC-DES, CBC-TDES, and the proposed algorithm will be tested under windows 98 operating system. Performance test will be performed on a personal computer (PC), which is equipped with Pentium III CPU (550 MHZ) and 256 MB of RAM. Visual C++ was used to implement these three algorithms. MAC generation will be executed on 1MB text file. In DES, key and IV size (Initialization Vector) are 64-bit each. TDES will use three keys with 64-bit size each. The proposed algorithm will use a key of 64-bit size.

Table 1 shows the results of execution time experiment. This time represents the time that a CPU spends completing necessary calculations for the three algorithms, and I/O operation. Encryption time for in the CBC-DES and CBC-TDES does not include "Convert-To-Binary" time. In addition, these results are normalized to CBC-DES processing time.

Table 1: CPU execution time (seconds)

Encrypting and MAC Generation			
File size	**CBC-DES**	**CBC-TDES**	**Proposed Algorithm**
0.5 MB	1	2.867	2.855
1 MB	1	2.897	2.878
2 MB	1	2.897	2.894

The results from the Table 1 show that the proposed algorithm is faster than CBC-TDES, and slower than CBC-DES. Because CBC-TDES calls DES algorithm three times. Moreover, these results show that the proposed algorithm, CBC-DES, and CBC-TDES are scalable. Besides algorithm structure, execution time is affected by key length.

Key length effect happens when running the same algorithm without any change to its internal structure, using different key lengths. In the proposed algorithm, there is a small change in time needed to run the proposed algorithm with an increase in a key length. Figure 2 shows an increase in key length by power of two with a little decrease in execution time, because we process a large data size each time rather than dealing with small block size. This figure uses a 1MB file.

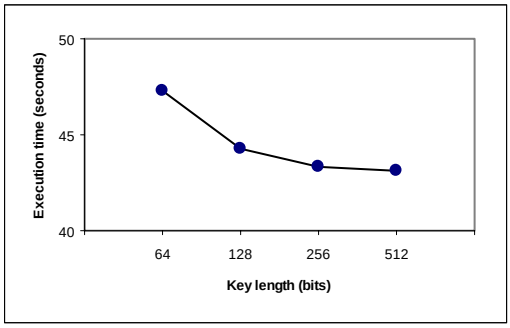

Figure 2: Key length effect

However, for CBC-DES algorithm and other encryption algorithm algorithms, running the same algorithm with different key lengths means a change in the algorithm structure and coding.

3.3 Key recovery attacks

Key recovery attack is based on trying all key values on few Message/MAC pairs to generate MAC values, then comparing all generated MAC's with message/MAC pair until the correct key is founded. In any message authentication scheme that uses a specified key length, this type of attack is theoretically possible. Such schemes are CBC-DES and CBC-TDES. However, if we take into account the required time and memory space, we need a lot of effort and highly equipped computers for this approach. The proposed algorithm does not limit key space and makes it application-dependant. However, it is preferred to be equal to the least message size used by the application. Therefore, previous explanations are theoretically valid for the proposed algorithm. However, the following reasons provide a resistance proof against key recovery attack:

1. The use of rotated secret S-box.
2. The use of long one-time key: trying all key space is not useful. Because after generating MAC for a message, key is changed to new one as lemma 2 explains.

Table 2 shows the needed key recovery time for the three algorithms; CBC-DES, CBC-TDES, and the proposed algorithm. All of them are assumed to work with an application that deals with messages of 1 MB long at least. Key length at the proposed algorithm is assumed to be 1KB long, 64-bit for CBC-DES, and 64-bit for each key of the three keys used with CBC-TDES. It shows that the time to recover the proposed algorithm key is greater that other algorithms. In addition, MAC size using the proposed algorithm is 256 bits (in this example).

Table 2: key recovery time

Algorithm	Key space	Worst attack Time	MAC-size
CBC-DES	2^{64}	$2^{64} * 16.5_{Sec}$	64 bits
	2^{128}	$2^{128} * 16.5_{Sec}$	128 bits
CBC-TDES	$3*2^{64}$	$3 * 2^{64} * 47.8_{Sec}$	64 bits
	$3*2^{128}$	$3 * 2^{128} * 47.8_{Sec}$	128 bits
Proposed-algorithm	2^{8192}	$2^{8192} * 47.5_{sec}$	> 256 bits

4. Conclusion

The design of the proposed algorithm considers speed improvements in the future; therefore, we decided to use large and variable key size. The result was the use of one-time key and variable message-based MAC length. This paper explains the differences between new developed algorithm and CBC. Finally, we listed some extracted properties for the proposed algorithm.

References

[1] ISO / IEC 9797. Data cryptographic techniques – Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm, 1989.

[2] ANSI X9.9, American National Standard for Financial Institution Message Authentication (Wholesale), American Bankers Association, 1981. Revised 1986.

[3] Krawczyk H. Personal Communication, September 1994.

[4] Preneel B, and Van Oorschot P. A new generic attack on message authentication codes. Advances in Cryptology, Crypto 95 Proceedings, Lecture Notes in Computer Science; D. Coppersmith ed; Springer-Verlag; 1995; 963.

[5] Campbell, C. Design and specification of cryptographic capabilities. In Computer security and the data encryption standard, NBS Special Publication 500-27; D. Barnstad, Ed; National Bureau of Standards, Washington, D.C., 1977; 54–66.

[7] M. Luby and C. Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions; SIAM J. Comput.; April 1988; 17(2).