World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:3, 2008

Use of Novel Algorithms MAJE4 and MACJER-320 for Achieving Confidentiality and Message Authentication in SSL & TLS

Sheena Mathew, K. Poulose Jacob

Abstract**—**Extensive use of the Internet coupled with the marvelous growth in e-commerce and m-commerce has created a huge demand for information security. The Secure Socket Layer (SSL) protocol is the most widely used security protocol in the Internet which meets this demand. It provides protection against eaves droppings, tampering and forgery. The cryptographic algorithms RC4 and HMAC have been in use for achieving security services like confidentiality and authentication in the SSL. But recent attacks against RC4 and HMAC have raised questions in the confidence on these algorithms. Hence two novel cryptographic algorithms MAJE4 and MACJER-320 have been proposed as substitutes for them. The focus of this work is to demonstrate the performance of these new algorithms and suggest them as dependable alternatives to satisfy the need of security services in SSL. The performance evaluation has been done by using practical implementation method.

Keywords**—**Confidentiality, HMAC, Integrity, MACJER-320, MAJE4, RC4, Secure Socket Layer

I. Introduction

WITH the growth of the Internet and digital transmission, many applications need to transmit data to remote applications and computers securely. The four main stated security issues are confidentiality, authentication, integrity and non-repudiation. Confidentiality means, only authorized users can access the information while unauthorized users are to be denied access. Authentication has to guarantee that the user accessing the information is truly the intended person and not a pretender. Integrity has to ensure that the received information is same as the transmitted information without being modified by others during transmission. Non-repudiation guarantees that senders and receivers have undeniably transmitted or received information, respectively. These four mentioned issues are interdependent and must therefore be addressed simultaneously in the design of security systems. SSL protocol has been universally accepted in the World Wide Web for authenticated and encrypted communication between clients and servers.

Sheena Mathew and K. Poulose Jacob are with the Department of Computer Science, Cochin University of Science and Technology, Kochi, Kerala, India. e-mail: sheenamathew@cusat.ac.in, kpj@cusat.ac.in

The SSL protocol was originally developed by Netsape, its version 1.0 was never publicly released; version 2.0 was released in 1994 but contained a number of security flaws which ultimately led to the design of version 3.0 which was released in 1996[1]. At present, SSL is widely deployed in many intranets as well as over the public Internet and has become the de facto standard for transport layer security. Recently, the Internet Engineering Task Force (IETF) has started an effort to standardize SSL as an IETF standard under the name of Transport Layer Security (TLS) protocol [2]. The few real world, practical applications of SSL & TLS are [3] client server systems, financial systems, information systems to create remote access and administration applications, travel industry to create online reservation systems and secure information transfer, etc. Visa, MasterCard, American Express and many leading financial institutions have endorsed SSL for commerce over the Internet. Some early implementations of SSL used 40-bit symmetric keys because of US government restrictions on the export of cryptographic technology. The 40-bit key size limitation has mostly gone away and modern implementations use 128-bit (or longer) keys for symmetric key ciphers.

One of the reasons that SSL has outgrown other transport and application layer security protocols such as SSH, SET, and SMIME in terms of deployment is that it is application protocol independent [4]. Conceptually, any application that runs over TCP can also run over SSL. There are many examples of applications such as TELNET and FTP running transparently over SSL. However, SSL is most widely used as the secure transport layer below HTTP. A large number of e-commerce sites dealing with private and sensitive information use SSL as the secure transport layer. This number is expected to grow, as more and more businesses and users embrace electronic commerce. As security becomes an integral feature of Internet applications and the use of SSL rises, its impact on the performance of the servers as well as the clients is going to be increasingly important. Browsers like Netscape Navigator and Internet Explorer can access SSL enabled web pages by using URLs that start with 'https' instead of 'http'.

The main objectives for SSL are:

1. Authenticating the client and server to each other.
2. Ensuring data integrity

3. Securing data privacy.

II. Motivation

In applications using SSL, the confidentiality of information is ensured using strong encryption algorithms. For very fast encryption and decryption of data for transmission after an SSL connection has been established, RC4 is the preferred algorithm. Similarly HMAC-SHA-1 has been recommended for message authentication in several network security protocols. The key reasons behind this are the free availability, the flexibility of chaining the hash function and the reasonable speed, among others. Even though RC4 and HMAC-SHA-1 are the most widely used ciphers of secure web applications, the strength of RC4 [5] and SHA-1 [6] has been called into question as a result of recent findings. Hence it is required to have proven and new methods to meet the future requirements. The analysis of novel cryptographic algorithms MAJE4 [7] and MACJER-320 and its performance in comparison with the popular RC4 and HMAC-SHA1 have been done in this context and the novel algorithms MAJE4 and MACJER-320 have been proposed as alternatives.

III. Description of existing algorithms

This section presents an overview of the cryptographic algorithms RC4 and HMAC-SHA-1 used in SSL.

A. RC4

RC4 is a variable key-size stream cipher developed in 1987 by Ron Rivest for RSA Data Security, Inc. The RC4 stream cipher has two phases, the key set-up and the keystream generation. Both phases must be performed for every new key. The algorithm is based on the use of a random permutation. A variable length key K, of size 1 to 256 bytes is used to initialize a 256-byte state vector S, with elements S_0, S_1, ….., S_{255}.

Initially the entries of S are set to the values 0 to 255 in ascending order. A temporary vector T, is also created. For a key of length keylen bytes, the first keylen elements of T are copied from K, and then K is repeated as many times as necessary to fill out T. Next, we use T to produce the initial permutation of S. The pseudo-code for the key setup is as follows:

```
for i = 0 to 255
    $S_i = i$
    $T_i = K[i \bmod keylen]$
endfor
    k=0
for i = 0 to 255
    $k = (k + S_i + T_i) \bmod 256$
    Swap($S_i$, $S_k$)
endfor
```

Once S is initialized, the input key is no longer used. The next phase is key stream generation which is described by the pseudo-code as:

```
i = 0
k = 0
while(true)
i = (i + 1) mod 256
$k = (k + S_i) \bmod 256$
Swap($S_i$, $S_k$)
$t = (S_i + S_k) \bmod 256$
key = $S_t$
endloop
```

For encryption, the value key is XORed with the next byte of plaintext. For decryption, the value key is XORed with the next byte of cipher text.

B. HMAC

The different variables used in the HMAC algorithm are shown in Table I.

TABLE I
BASIC NOTATIONS IN HMAC

MD	Message digest/ hash function
M	Input message
B	Number of bits in each block
K	Shared symmetric key
K1	Transformed key K1
Ipad	String 00110110 repeated b/8 times
Opad	String 01011010 repeated b/8 times
H	Hash code

The step-by-step approach of the HMAC message authentication code is given in Table II.

TABLE II
HMAC ALGORITHM

Step 1:	Make the length of K equal to B. Append enough zeros to the left end of K to create a B bit key K1.
Step 2:	XOR K1 with Ipad to produce the B bit block S.
Step 3:	Append M to S. That is the original message is simply appended to the end of S.
Step 4:	Apply the Message digest algorithm/ Hash function to the output of Step 3 to produce hash code H.
Step 5:	XOR K1 with Opad to produce the B bit block S1.
Step 6:	Append the hash code H produced in Step 4 to S1.
Step 7:	Apply the message digest algorithm/ Hash function to the output of Step 6 to produce the final MAC.

C. SHA-1

The Secure Hash Algorithm (SHA) [8] was developed by National Institute of Standards and Technology (NIST) along with NSA.

IV. Description of new algorithms

This section presents the novel MAJE4 stream cipher algorithm and MACJER-320 algorithm along with JERIM-320 hash function [9] for achieving confidentiality as well as message authentication.

A. MAJE4

a. Main features of MAJE4

1. The encryption sequence can have a large period.
2. The key stream can approximate the properties of a true random stream.
3. MAJE4 is suitable for hardware or software and it uses only primitive computational operations commonly found in microprocessors.

4. It is simple and fast. It uses simple algorithm, which is easy to implement and eases the task of determining the strength of the algorithm.
5. Low memory requirement makes it suitable for handheld type devices with restricted memory.
6. Mixed operators are used for the design of MAJE4. The use of more than one arithmetic and / or Boolean operator complicates cryptanalysis. Primitive operators like + and ^ are used since these operators do not commute and hence cryptanalysis becomes more difficult.
7. It should have a flexible security choice with the key sizes of 128 or 256 bits.

b. Key setup of MAJE4

One can choose either 128 or 256 bit key for the algorithm given in Table III.

128-bit key: The four 32 bit words, ie. $key_{[0]}$, $key_{[1]}$, $key_{[2]}$ and $key_{[3]}$ are considered for storing the key.

256-bit key: The key is stored in eight 32 bit words $key_{[0]}$, $key_{[1]}$, $key_{[2]}$, $key_{[3]}$, $key_{[4]}$, $key_{[5]}$, $key_{[6]}$ and $key_{[7]}$.

TABLE III
ALGORITHM OF MAJE4

```
Step 1:  Assign the key length kl either as 128-bit or 256-bit.
Step 2:  if kl = 128 then
             kln = 2, div = 4
         else
             kln = 3, div = 8
Step 3:  if kl = 128 then consider two lsb's of $key_{[0]}$ and find its decimal
             equivalent and store in the variable 'in'.
         else
             if kl = 256 then consider three lsb's of $Key_{[0]}$ and find its
             decimal equivalent and store in the variable 'in'.
Step 4:  ran = $key_{[0]}$ ^ $key_{[in]}$
Step 5:  if kl = 128 then consider two lsb's of ran and find its decimal
             equivalent and store in the variable 'in1'.
Step 6:  if kl = 256 then consider three lsb's of ran and find its decimal
             equivalent and store in the variable 'in1'.
Step 7:  check the 16th bit in ran,
         if it is 1 then
             newran = ($key_{[in1]}$ + $key_{[in1+1 mod div]}$) ^ ($key_{[in1+2 mod div]}$ + $key_{[in1+3 mod div]}$)
         else
             newran = ($key_{[in1]}$ ^ $key_{[in1+1mod div]}$) + ($key_{[in1+2 mod div]}$ ^ $key_{[in1+3 mod div]}$)
Step 8:  The output 32-bit word is newran, which can be used to XOR
             with the corresponding word in the plain text.
Step 9:  Advance all the keys as
         $key_{[i]}$ = $key_{[i]}$ * $key_{[i]}$ + $key_{[i]}$ >> 20
Step 10: go to step3
```

B. MACJER-320

The variables used in the MACJER-320 construction are given in Table IV.

TABLE IV
VARIABLES USED IN MACJER-320

K	Shared symmetric key.
M	Input message.
B	Number of bits in each block.
MDA	Message digest algorithm or Hash Function (JERIM-320)
H	Hash code
SH	Circular shifted hash code H
SK	Circular shifted key K

The step-by-step approach of MACJER-320 is given in Table V.

TABLE V
MACJER-320 ALGORITHM

Step 1: Make length of K equal to B.
Here the initial key K is 320-bit long and the block length B is 512-bit. To make the length of K equal to the block length add as many 0 bits as required to the left of K. Hence add 192, 0 bits to the left of key K.

Step 2: Prefix and suffix the key along with the message.
Divide the key in to two equal parts (256 bits each), and then prefix the message using 256 lsb bits of the key and suffix the message using 256 msb bits of the key.

Step 3: Apply the message digest algorithm / hash function
Now, JERIM-320 is applied to the output of step 2 (i.e. to the combination of the 256 lsb bits of the key, the message, the 256 msb bits of the key) to produce the 320-bit hash code H.

Step 4: Circular shift hash code H and the initial key K
Circular shift H by 13 bits and key K by 17 bits to the left to produce the shifted hash SH and the shifted key SK.

Step 5: XOR K with SH to produce KSH
Now XOR K with SH to produce a variable called as KSH.

Step 6: Add H with SK to produce HSK
Now add H with SK to produce a variable called as HSK

Step 7: XOR KSH with HSK to produce MAC
XOR KSH with HSK to produce the final 320-bit message authentication code.

C. JERIM-320

a. Structure of JERIM-320

Fig.1. Shows the outline of the compression function of JERIM-320. It consists of four parallel branches B1, B2, B3 and B4.

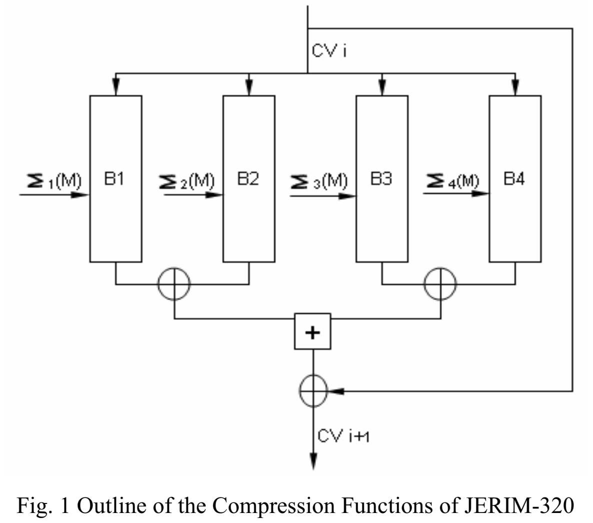

Fig. 1 Outline of the Compression Functions of JERIM-320

The initial chaining variable CV_i is given as input to the compression functions. CV_i consists of 10 registers A,B,C,D,E,F,G,H,I and J.

Each successive 512-bit message block M is divided into sixteen 32 bit sub blocks M_0, M_1, …, M_{15} given as $\Sigma_i(M)$ as input to all four branches and a computation is done to update CV_i to CV_{i+1} as

$CV_{i+1}=CV_i$^ ((B1output ^ B2output) + (B3output ^ B4output)). Finally the message is transformed into the 320-bit hash value.

b. Single Step Operations

Five rounds are used in JERIM-320 for each 512-bit message block. The sixteen 32-bit sub blocks of the 512-bit block in each round are processed in four parallel branches. The inputs to each single step operations are the sixteen sub blocks, the chaining variables A1, B1,…J1, A2, B2, …J2, A3, B3,….J3, A4, B4,…..J4 of each branch and the constants $K_{[t]}$. Order of message words, shift values, Boolean functions and constants in each branch and each round are different. There are 16 single step iterations in each round and in all the four branches as shown in Fig. 2. The output of each iteration is copied again into the chaining variables A1, B1,…J1; A2, B2, …J2; A3, B3,….J3; A4, B4,…..J4 and so on.

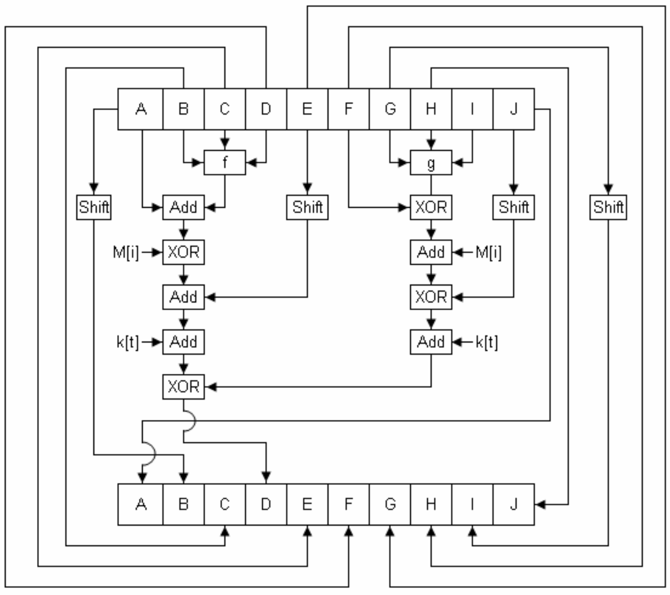

Fig. 2 A Single Step Operation of JERIM-320

V. Security analysis

Sections A and B describe an analysis of the stream ciphers RC4 and MAJE4. The security analysis of HMAC is explained in section C. Sections D and E describe an analysis of the properties of Message Authentication Code and Hash Function, which help MACJER-320 to achieve a significantly higher level of security than the popularly used ones.

A. RC4

Some of the published attacks on RC4 are as follows:

1. The first known weaknesses in RC4 were reported in 1995 by Ross [10] and Wagner [11]. They described several classes of keys that have specific weaknesses including predictable output or output that leaks key information. Later a related key attack was observed for long keys (2048 bits) [12].
2. Since the output of RC4 stream cipher is used to encrypt the plain text by bitwise XOR, any observable bias in the output can be used as the basis for an attack. A correlation was detected by Golic [13] between bytes at time t and t+2. Many stronger correlations were later reported by Fluhrer and McGrew [14].
3. Attacks to guess the internal state and then check for consistency with known output have been studied independently by several researchers and the results were published [15]-[17].
4. The most significant attacks on RC4 have been based on exploiting the simplicity of the initialization algorithm to discover an observable bias in the first few bytes of the output sequence. A bias in the second output byte also has been reported [18]. The value zero occurs with twice the expected probability for a random sequence. A bias in the first byte was also reported [19].
5. S. Fluhrer, I. Martin and A. Shamir published a report [5] that describes several weaknesses in the key scheduling algorithm of RC4 and proposes attacks for exploiting those weaknesses.
6. Klein [20] showed an improved way of attacking RC4 using related keys that does not need the 'resolved condition' on the IVs and gets by with a significantly reduced number of frames.
7. Subhamoy Maitra and Goutam Paul gave an independent analysis [21] of Klein's attack with results similar to our multiple key bytes extension.
8. Vaudenay and Vuagnoux presented a similar attack at SAC2007 [22], which additionally makes use of the fact that the RC4 key is stretched to 256 bytes by repeating it. The same trick was reported by Ohigashi, Ozasa, Fujikawa, Kuwadako and Morii [23], who developed an improved version of the attack.
9. The implication of these findings is that a buffer overflow attack or a similar attack can be used to learn a single state of the generator, which can then be used to predict all random values, such as SSL keys [24]. This type of attack is more severe and more efficient than other known attacks.

These problems with RC4 have raised fears on the security of protocols like the SSL which are using RC4 for providing confidentiality.

B. MAJE4

The MAJE4 is a 128-bit or 256-bit key algorithm and the randomness property of the stream cipher is analyzed by using the five statistical tests like frequency test, serial test, poker test, runs test and autocorrelation test [24]. All the five statistical tests are passed by this generator for all the random streams produced. Hence MAJE4 algorithm can be used very well for encrypting the message of any length.

C. HAMC-SHA-1

1. In HMAC the XOR with Ipad results in flipping one-half of the bits of K. Similarly the XOR with Opad result in flipping the other-half of the bits of K, but a different set of bits. In effect, by passing S and S1 through the compression function of the hash algorithm, we have pseudo randomly generated two keys from K, which add security to HMAC.

2. The recent attacks on Wang et al and Biham et al. have undermined the confidence in the popular hash functions such as MD5 or SHA-1.
3. As outlined in the paper "keying hash functions for message authentication", HMACs can be vulnerable to birthday, collision and other attacks [25].
4. Other publications "On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1" [26] and "Note on Distinguishing, Forgery and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entrophy of NMAC" [27] have shown how to use the differential distinguishers to devise a forger attack on HMAC.
5. The strongest attack known against HMAC is based on the frequency of collisions for the hash function. With this, HMACs have become more insecure [28].
6. The attack Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu recently announced on SHA-1 indicates that the algorithm isn't quite as strong as it was thought to be, as it takes only 2^{69} steps to find a collision instead of the expected 2^{80} steps.

D. MACJER-320

1. The security of the message authentication mechanism presented here depends mainly on the cryptographic properties of the hash function JERIM-320 as mentioned in section E
2. The length in bits of a message authentication code is directly related to the number of trials that an intruder has to perform before a message is accepted. For a MAC value of bit-length m, the intruder has to perform on average 2^{m-1} random on-line MAC verifications before his strategy succeeds. Thus in MACJER-320, an intruder requires 2^{320-1} trials.
3. The message is enveloped with a secret prefix and a secret suffix before the hash code is computed. This hybrid method is stronger than either the prefix or the suffix variant [29] and provides protection against message substitution attacks when used in conjunction with a strong hash function JERIM-320. Also the splitting of the key into two parts strengthens the key by increasing confusion at the cipher text level [30].
4. Another important property of this hybrid method is its resistance to birthday attacks [31]. Consideration of these attacks is important since they strongly improve on exhaustive search attacks. Since these attacks require knowledge of the MAC value (for a given key) on about $2^{n/2}$ messages (where n is the length of the hash output) for values of $n \geq 320$ the attack becomes totally infeasible.
5. When combining functions and operations together, orthogonal operations like exclusive or and addition are used to create confusion and diffusion in the MAC.
6. The shifting of the hash code and key was done to increase confusion thus strengthening the output.
7. XORing has the effect of randomizing the input almost completely and overcoming any regularity that appear in the output.

E. JERIM-320

1. The main difficulty in cryptanalyzing JERIM-320 comes from the fact that the same message blocks are given as input to each of the four streams in a permuted fashion. The attacker who tries to break JERIM-320 should aim simultaneously at four ways where the message difference passes, which would make the attacks more difficult.
2. By using one message block twice at each single step, it has been made difficult to construct a differential characteristic with high probability.
3. To avoid an attack that depends on brute-force methods, the output from the hash function has been made sufficiently long.
4. While combining the outputs from the four branches, orthogonal operations (+ and ^) are used to create confusion and diffusion which adds to the security.
5. There is a strong avalanche effect; hence a change in a single message bit affects all the registers after five rounds.
6. All shortcut attacks on MD5 target one of the intermediate blocks. Increasing the intermediate value to 320 bits helps to prevent these attacks.
7. The single step operation ensures that changing a small number of bits in the message affects many bits during the various passes. Together with the strong avalanche, helps JERIM-320 to resist attacks similar to Dobbertin's differential attack [32] on MD4.

It is in this situation that MAJE4 and MACJER-320 have been proposed along with a strong hash function JERIM-320 for achieving better security services.

VI. Performance evaluation

The performance evaluation is done by comparing RC4 with MAJE4 and MACJER-320 with HMAC-SHA1 as given in section A and section B. The evaluations were done using Pentium IV processor, Linux operating system and C compiler.

A. MAJE4 & RC4

From the timing analysis it can be noted that when we compare RC4 and MAJE4, MAJE4 is almost 1.2 times faster as shown in Table VI. On comparing the memory required for executable files of RC4 and MAJE4, MAJE4 was found consuming lesser space compared to RC4. The memory size required for optimised code for RC4 is 8077 bytes and for MAJE4 is 5435 bytes.

TABLE VI
TIMING ANALYSIS & MEMORY REQUIREMENT

PRNGs	MAJE4	RC4
Key length	128-bit	128-bit
No. of random numbers generated	1,15,39,399	3,95,99,988
No. of random bits per each random number	32	8
Total no. of bits produced (speed Mbps)	352.15	302.12
Memory requirement (Bytes)	5435	8077

B. MACJER-320 & HMAC-SHA-1

The total number of operations, memory requirements and the speed performance of MACJER-320 using JERIM-320 hash function and HMAC using SHA-1 hash function, were compared. MACJER-320 produces a MAC of 320 bits where as HMAC-SHA-1 produces a MAC of 160 bits only. Hence MACJER-320 can definitely provide added security than HMAC-SHA1.

As shown in Table VII the total number of operations used in MACJER-320 is 3.7 times than that in HMAC-SHA1. The hash function JERIM-320 in MACJER-320 makes use of four parallel lines of message processing and hence the variables and computations required in JERIM-320 will be more compared to the hash function SHA-1 in HMAC. The multiple operations on the message blocks in MACJER-320 will result in much higher security with a negligible compromise in the speed of operation.

TABLE VII
COMPARISON BETWEEN MACJER-320 AND HMAC IN TERMS OF THE NUMBER OF OPERATIONS

Operation	MACJER-320 using JERIM-320	HMAC using SHA-1
Addition	46	24
Bitwise operation(^,V, Λ,¬)	193	39
Shift operation	41	13
Total number of operations	280	76

As shown in Table VIII the memory requirement for MACJER-320 is more than that of HMAC-SHA1 and the speed of MACJER-320 is less than that of HMAC-SHA1. These are because of the increased number of Boolean functions, the need for other operations like add, shift as well as the greater number of lines of message processing used in JERIM-320 than in SHA-1. Even though the speed of MACJER-320 is less than that of HMAC-SHA-1, it is very much within the acceptable limits and hence the advantages due to increase in the security overcomes the reduction in speed.

TABLE VIII
PERFORMANCE COMPARISONS BETWEEN MACJER-320 AND HMAC

Algorithm	Speed (Mbps)	Memory requirement (Bytes)
MACJER-320IM using JERIM-320	13.15	12530
HMAC using SHA-1	57.58	8074

VII. Conclusion

The SSL designers have chosen to use the then available algorithms RC4 as fast stream cipher and HMAC as hash-based construction for its security services. But few recent findings show that the confidence level in these algorithms is coming down. It is clear that a transition to a newer encryption and message authentication algorithms will be required in the near future, since the information handled is very sensitive. It is in this situation that more secure algorithms MAJE4 and MACJER-320 are suggested which can definitely become good substitutes.

References

[1] Transport layer Security, Wikipedia, http://en.wikipedia.org/wiki/Secure_Sockets_Layer

[2] C.Allen and T.Dierks, The TLS Protocol Version 1.0. Internet Draft, Internet Engineering Task Force, November 1997, http://tools.ietf.org/html/rfc2246

[3] Security Protocols Overview An RSA Data Security Brief, www.comms.scitech.susx.ac.uk/fft/crypto/security-protocols.pdf

[4] George Apostolopoulos, Vinod peris, Prashant Pradhan, Debanjan Sahi, "Securing Electronic Commerce: Reducing the SSL Overhead", IEEE Network, 14(4) : pp. 8-16, July 2000.

[5] S. Fluhrer, I. Mantin, A. Shamir, " Weakness in the key scheduling Algorithm of RC4", Proceedings in the selected Areas in Cryptography 2001, SAC'01, LNCS vol.2259, pp. 1-24, Springer-Verlag, 2001.

[6] Xiaoyun Wang and Hongbo Yu, "How to break MD5 and other hash functions", Advances in Cryptology – EUROCRYPT, LNCS 3494, Springer-Verlag , pp.19-35, 2005.

[7] Sheena Mathew, K.Paulose Jacob, "A New Fast Stream Cipher: MAJE4", Proceedings of IEEE, INDICON 2005, pp60-63, 2005.

[8] National Institute of Standards and Technology (NIST) (2002), FIPS-180-2: Secure Hash Standard, at http://csrc.nist.gov/publications/fips/fips 180-2/fips 180-2.pdf.

[9] Sheena Mathew, K. Poulose Jacob, "JERIM-320: A New 320-bit Hash Function with Higher Security", International Journal of Computers, Systems and Signals, to be published.

[10] A.Roos, "A Class of weak keys in the RC4 stream cipher", sci.crypt, 1995.

[11] D.Wagner, " My RC4 weak keys", sci.crypt, September 1995.

[12] A.I.Grosul and D.S.Wallach, "A Related Key Cryptanalysis of RC4", Manuscript from Department of Computer Science, Rice University, 6 June 2000.

[13] J.Dj.Golic, "Linear statistical Weakness of alleged RC4 keystream generator", Advances in Cryptology – Eurocrypt 97, LNCS vol. 1233, pp.226-238, Springer-Verlag, 1997.

[14] S.R.Fluhrer and D.A.McGrew, "Statistical Analysis of the Alleged RC4 Keystream Generator", Proceedings of Fast Software Encryption 2000, LNCS vol. 1978, pp.19-30, Springer-Verlag, 2001.

[15] S.Mister and S.E.Tavares, "Cryptanalysis of RC4-like Ciphers", Proceedings of SAC'98, LNCS vol. 1556, pp.131-143, Springer-V0000erlag, 1999.

[16] L.Knudsen, W.Meier, B.Preneel, V.Rijmen and S.Verdoolaege, "Analysis methods for (alleged) RC4", Advances in Cryptology – AsiaCrypt 98, LNCS vol.1514, pp.327-341, Springer -Verlag, 1998.

[17] J.Dj.Golic, "Iterative Probabilistic Cryptanalysis of RC4 Keystream Generator", Proceedings of ACISP 2000, LNCS vol.1841, pp. 220-233, Springer – Verlag, 2000.

[18] I.Mantin and A. Shamir, " A Practical Attack on Broadcast RC4", Proceedings of Fast Software Encryption, 2001, LNCS, vol.xx, pp.152-164, Springer-Verlag, 2002.

[19] I.Mironov, "(Not so) Random Shuffles of RC4", Advances in Cryptology –CRYPTO-2002, LNCS vol.2442, pp. 304-319, Springer Verlag, 2002.

[20] Andreas Klein, "Attacks on the RC4 stream cipher", Designs, Codes and Cryptography, 2007

[21] Subhamoy Maitra and Goutam Paul, "Many keystream bytes of RC4 leak secret key information", Cryptology ePrint Archieve, Report 2007/261, 2007, http://eprint.iacr.org/.

[22] Serge Vaudenay and Martin Vuagnoux, Passive-only key recovery attacks on RC4. In Selected Areas in Cryptography 2007, Lecturer Notes in Computer Science, Springer 2007

[23] Toshihiro Ohigashi, Hidenori Kuwakado, and Masakatu Morii, "A Key recovery attack on WEP with less packets", Technical Report of IEICE, ISEC Nov., 2007

[24] D.E.Knuth, *The Art of Computer Programming*, Vol.2, Seminumerical Algorithms, Third Edition, Addison – Wesley, 1997.

[25] Mihir Bellare, Ran Canetti, Hugo Krawczyk (1996), "Keying Hash Functions for Message Authentication", *Advances in Cryptology-CRYPTO*, LNCS 1109, Springer- Verlag, pp 1-15.

[26] Jongsung Kim, Alex Biryukov, Bart Preneel, Seokhie Hong (2006), "On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0 and SHA-1", Proceedings of SCN, LNCS 4116, Springer-Verlag, pp 242-256.

[27] Christian Rechberger and Vincent Rijmen, "Note on Distinguishing, Forgery, and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entropy of NMAC", 2006, URL: http://citeseer.ist.psu.edu/cache/papers/cs2/338/http:zSzzSzeprint.iacr.orgzSz2006zSz290.pdf/note-on-distinguishing-forgery.pdf

[28] Mihir Bellare, Ran Canetti, Hugo Krawczyk (1996), "Message Authentication using Hash Functions the HMAC Construction, CryptoBytes, Vol 2, No.1, RSA Laboratories pp 1-5.

[29] Gene Tsudik (1992), "Message Authentication with One-Way Hash Functions", Proceedings of IEEE-INFOCOM, pp 2055-2059.

[30] Thomas Calabrese (2006), "Information Security Intelligence Cryptographic Principles and Applications", Thomson Delmar Learning, India.

[31] Wagner D., "A Generalized Birthday Problem", Proceedings of Crypto '02, LNCS vol. 2442, Springer-Verlag, 2002.

[32] H. Dobbertin (1996) "Cryptanalysis of MD4", *Fast Software Encryption*, LNCS 1039, Springer-Verlag, 53-69.

Sheena Mathew, Reader in Cochin University of Science and Technology (CUSAT), Kochi, Kerala, India has 15 years of teaching experience in Computer Science. She had her graduation from Madurai Kamaraj University and post graduation from the Indian Institute of Science, Bangalore. She is presently a research scholar; her areas of interest being Cryptography and Network Security. She has 8 publications in various international journals and conferences to her credit.

Dr. K. Poulose Jacob, a National Merit Scholar all through, got his degree in Electrical Engineering in 1976 from University of Kerala, followed by his M.Tech. in Digital Electronics and Ph. D. in Computer Engineering from CUSAT, Kochi. He has been teaching at CUSAT since 1980 and currently occupies the position of Professor and Head of the Department of Computer Science. He has served as a Member of the Standing Committee of the UGC on Computer Education and Development. He is on the editorial board of two international journals and has more than 60 papers in various international journals and conferences to his credit. His research interests are in Information Systems Engineering, Intelligent Architectures and Networks.