Improvement in Message Authentication Algorithm using TLS1.0

Prajakta Gandhe [1] Dhruv Gupta [2] Harshini Pathak [3]
[1, 2, 3] TCET , Mumbai

Abstract---The main focus of this project is to enhance the data integrity in next generation encryption using TLS1.0. Our aim is to remove the pre-existing vulnerabilities in the present SHA algorithm and make it more attack resistant. TLS 1.0 supports encryption, authentication and key exchange. This project presents an enhancement (SHA-176) in the authentication of the existing authentication algorithm (SHA-160). The current algorithm used in TLS for message authentication and data integrity is mainly Secure Hashing Algorithm (SHA). This project tries to enhance the current SHA-160 algorithm so as to make the algorithm more attack resistant. The current SHA algorithm uses a 160 bit message digest but is susceptible to certain attacks; it has helped in coming over the vulnerabilities of MD5 but has certain problems which have to be overcome. This project presents an enhancement in the algorithm by forming a 176 bit message digest thus increasing the size of the digest as compared to the original algorithm. The SHA-176 uses 11 chaining variables of 16 bit each consisting of 80 rounds. The proposed algorithm has proved to increase security for the hashing techniques so that the user feels safe while transmitting data over the transport layer. The hashing times differ by a small factor but we can still use the SHA-176 model in TLS 1.0 for Message authentication as the security in SHA-176 is more and it is a stronger algorithm as compared to SHA-160.

I. Introduction

Secure communication is when two entities are communicating and do not want a third party to listen in. For that they need to communicate in a way not susceptible to eavesdropping or interception. Secure communication includes means by which people can share information with varying degrees of certainty that third parties cannot intercept what was said. Other than spoken face-to-face communication with no possible eavesdropper, it is probably safe to say that no communication is guaranteed secure in this sense, although practical obstacles such as legislation, resources, technical issues (interception and encryption), and the sheer volume of communication serve to limit surveillance. With many communications taking place over long distance and mediated by technology, and increasing awareness of the importance of interception issues, technology and its compromise are at the heart of this debate. For this reason, this article focuses on communications mediated or intercepted by technology.

TLS is a network layer protocol that is used to provide message encryption and authentication between applications and servers where data is send through insecure channel. TLS was released in response to the Internet community's demands for a standardized protocol. When a server and client communicate, TLS ensures that no third party may eavesdrop or tamper with any message. TLS is the successor to the Secure Sockets Layer (SSL) .TLS provides Message Authentication, Message Confidentiality and Key Exchange.

Message Authentication is the act of confirming the truth of an attribute of a datum or entity. This might involve confirming the identity of a person or software program, tracing the origins of an artifact, or ensuring that a product is what its packaging and labeling claims to be. Authentication often involves verifying the validity of at least one form of identification.

II. Background

A. Literature Survey

(1) International Journal of Computer Applications(0975-8887) Title is New Modified 256-bit MD5 algorithm with SHA Compression Function.

- Description MD5 256 bit message digest along with SHA compression function is used. The Hash value can be increased to 512, 768, etc. by increasing block size of compression functions or increasing the number of blocks.

(2) IJRIT International Journal of Research in Information Technology. Title is An approach for high security by using efficient SHA-176.

- Description: SHA-176 is used to remove the collision attacks found in SHA-160 and to make it resistant to all attacks.

(3) International Journal of Computer Technology and Electronics Engineering(IJCTEE). Title is Performance Analysis of SHA Algorithms: A Review.

- Description: SHA-192 algorithm is proposed which is an enhancement in SHA-160.The enhancement is to increase one chaining variable, i.e. to add an extra chaining variable F.

(4) IOSR Journal of Computer Engineering(IOSRJCE) Title is Evolution of SHA-176 Algorithm.

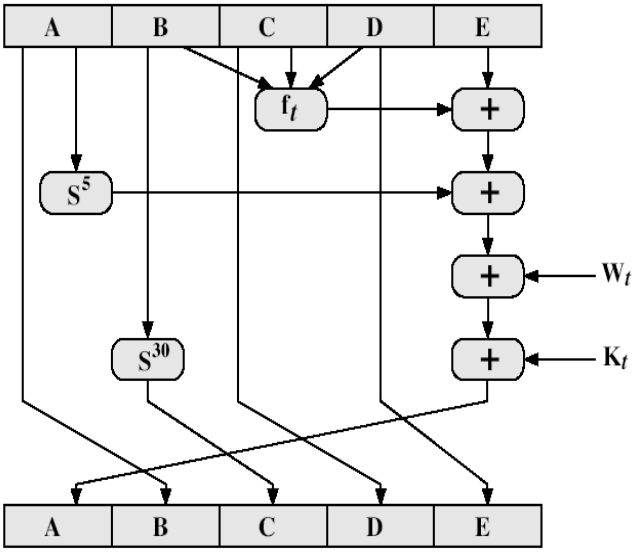
- Description: Easy to compute the hash value of any input. Input bits of any length will be hashed to 176 bit of fixed length.

B. SHA-160

SHA was designed by NIST & NSA in 1993, revised in 1995 as SHA-1.The US standard for use with DSA signature scheme are FIPS 180-1 1995, also Internet RFC317. The algorithm is SHA; the standard is SHS.SHA produces 160-bit hash values. It is now the generally preferred hash algorithm. The design of SHA is based on the design of MD4 with key differences.

A retronym applied to the original version of the 160-bit hash function published in 1993 under the name "SHA". It was withdrawn shortly after publication due to an

undisclosed "significant flaw" and replaced by the slightly revised version SHA



III. MODIFIED ALGORITHM

A. Proposed Model

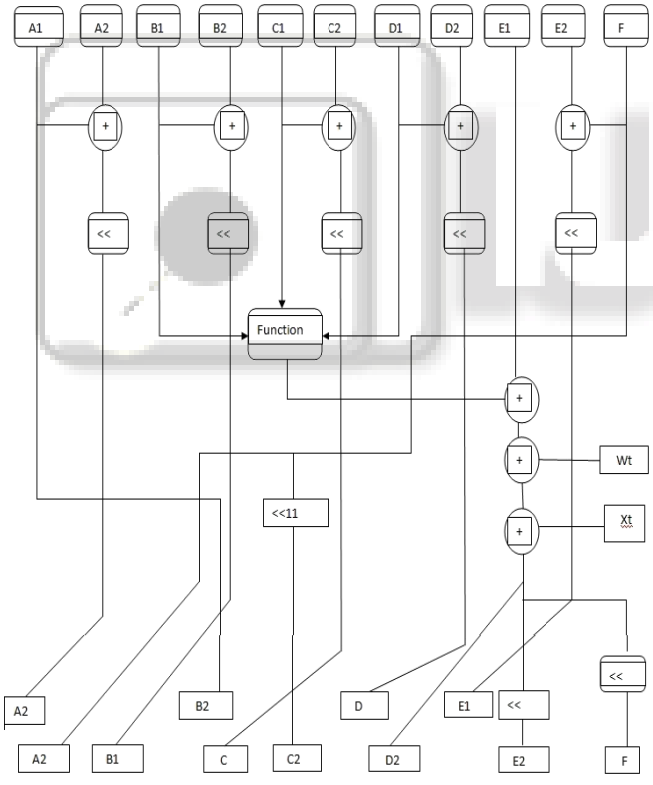


Fig. 2: Proposed Model

Steps of Algorithm are as follows:

Step 1: Padding

The first step in SHA-176 is to add padding bits to the original message. The aim of this step is to make the length of the original message equal to a value, which is 64 bits less than an exact multiple of 512. We pad message M with one bit equal to 1, followed by a variable number of zero bits.

Step 2: Append Length

After padding bits are added, length of the original message is calculated and expressed as 64 bit value and these 64 bits are appended to the end of the resultant message of step 1.

Step 3: Divide the input into 512 bit blocks

Divide the input message into blocks, each of length 512 bits, i.e. cut M into a sequence of 512 bit blocks $M^1,M^2…..M^N$. Each of M^1 parsed into thirty two 16 bit words $M^1…..M^{32}$.

Step 4: Initialize Chaining Variables

Before the hash function begins, the initial hash value H must be set. The hash is 176 bits used to hold the intermediate and final results. Hash can be represented as eleven 16 bit word registers A,B,C,D,E,F,G,H,I,J,K. The initial values of the chaining variable are:

A = 6745

B = 2301

C = EFCD

D = AB89

E = 98BA

F = DCFE

G = 1032

H = 5476

I = C3D2

J = E1F0

K = 4038

The compression function maps 176 bits value H=(A,B,C,D,E,F,G,H,I,J,K) and 512 bit block Mi into 176 bits value. The shifting of some of the chaining variables by 11 bits in each round will increase the randomness in bit change in the next successive routines. If the minimum distance of the similar words in the sequence is raised then the randomness will significantly raises. A different message expansion is employed in this hash function in such a way that the minimum distance between the similar words is greater compared with existing hash functions

Step 5: Processing

After pre-processing is completed each message block is processed in order using following steps:

I) For i = 1 to N prepare the message schedule.

Mit , $0 \le t \le 31$

Wt = (Wt-6 Wt-16 Wt-14 Wt-32) << 1

II) Initialize the eleven working variables A11,A12,B11,B12,C11,C12,D11,D12,E11,E12,F with (i-1)st hash value.

III) For t = 0 to 79

{

A11 = S2 (A11 XOR A12);

A12 = F;

B11 = S2 (B11 XOR B12);

B12 =A11;

C11 = S2 (C11 XOR C12);

C12 = S11 (F);

D11 = S2 (D11 XOR D12)

D12 = E11 XOR Funct (B11,C11,D11) XOR Wt XOR Kt

E11 = S2 (E11 XOR E12)

E12 = S8 (E11 XOR Funct(B11,C11,D11) XOR Wt XOR Kt)

F = S2 (E11 XOR Funct(B11,C11,D11) XOR Wt XOR Kt)

}

Where Kt is a constant , F1 is a bitwise Boolean function, for different rounds defined by,

Funct (B11, C11, D11) = IF B11 THEN C11 ELSE D11

Funct (B11, C11, D11) = B11 XOR C11 XOR D11

Funct (B11, C11, D11) = MAJORITY (B11, C11, D11)

Funct (B11, C11, D11) = B11 XOR C11 XOR D11

Where the "IF….THEN……ELSE "function is defined by

IF B11 THEN C11 ELSE D11 = (B11ΛC11) V ((¬B11) ΛD11)

And "MAJORITY" function is defined by

MAJ (B11, C11, D11) = (B11ΛC11) V (C11ΛD11) V (D11ΛB11)

Here, S2, S8 and S11 are left shift of 2 bit, 8 bit and 11 bit respectively

IV) H01 (i) = A11 + H01 (i-1)

H02 (i) = A12 + H02 (i-1)

H11 (i) = B11 + H11 (i-1)

H12 (i) = B12 + H12 (i-1)

H21 (i) = C11 + H21 (i-1)

H22 (i) = C12 + H22 (i-1)

H31 (i) = D11 + H31 (i-1)

H32 (i) = D12 + H32 (i-1)

H41 (i) = E11 + H41 (i-1)

H42 (i) = E12 + H42 (i-1)

H51 (i) = F + H5 (i-1)

IV. Scope of the Project

- Being an algorithm with larger message digest size that SHA-160 the algorithm will act as an efficient and reliable hashing technique.
- This model is also free from collision attacks where SHA-160 fails as it is prone to this type of attack.
- The project compares the performance and efficiency of SHA-160 and SHA-176.
- The time taken for hashing the SHA-176 digest can be reduced to make it more efficient.

V. Problem Definition

Security for data transmission is a necessity to counter act efforts of unauthorized data access and tampering. Many of these transmitted messages contain data which if intercepted could result into critical losses in many aspects.

The SHA algorithm was designed to overcome the weaknesses of the MD5 algorithm. Hashing is a one-way function that doesn't allow the reverse process. Thus they are more secure than other cryptographic algorithms .they are used to provide integrity over the content being transmitted.

But it was found the hash digest was prone to collision attacks. The hash value could never be found out but the message could be changed to generate the same hash value thus the receiver never came to know of the modification. This was possible in 2(L/2) attempts.

Thus there was a need to minimize the collision attacks on SHA-160 algorithm. This led to the development of the idea to develop the SHA-176 algorithm.

The SHA-176 algorithm has a larger message digest size and is not very easily attacked and is more powerful than SHA160.

Further the tests carried out in the project provide a better understanding of the performance of the SHA-176 as compared to SHA-160.

VI. Implementation

Stage 1: Implementation of AES Algorithm

Stage 2: Implementation of SHA 160 Algorithm

Stage 3: Implementation of SHA 176 Algorithm

Stage 4: Client-Server Channeling.

The results of the project have been formulated by running the program files on the following two configurations.

A. Configuration 1:

Processor: Pentium(R) Dual-Core CPU T4400 @2.20GHz 2.20GHz
Installed Memory (RAM): 2.00 GB
System Type: 64-bit Operating System

B. Configuration 2:

Processor: Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz 2.50GHz
Installed Memory (RAM): 4.00 GB (3.87GB usable)
System Type: 64-bit Operating System , x64-based processor

VII. Results

This section is providing analysis of presented algorithm on the basis of different parameters like security and time efficiency for encryption/decryption algorithm, java implementation has been used to test these algorithms.

A. Comparison of Hashing Algorithms:

The hashing algorithms SHA-160 and SHA-176 were tested based on the security and time needed to generate message digests for the text data. Based on the testing results, it was found that SHA-176 needs more time to generate a message digest when compared with SHA-160 because the message digest generated by the proposed algorithm is longer than the existing SHA-160. Hence the security of the existing algorithm gets improved. Thus the time to break 176 bit message digest will be more when compared with the

existing SHA-160. When comparing the bit difference, it is found that the bit difference in SHA-176 is more, after changing a single word in the message, as compared SHA-160. Table 6.1 is showing bits wise comparisons between existing SHA and SHA-176. Table 6.2 is showing execution time comparison between SHA-160 and SHA-176.

Algorithms	Hashing Time	Message Digested	
SHA 160	9.2ms	0db19a43 abf50458 f9474b74 458309cb cfe30450	5 words of 32 bits each
SHA 176	9.4ms	3cf2 5061 9b1e 6633 1f27 0641 f21a d709 ba93 ba38 4b56	11 words of 16 bits each

Table. 1: Timing Comparison between Existing SHA-160 and SHA-176 algorithms for a 3.04KB file.

Message	SHA 160	SHA 176
the lazy fox jumped over a dog	74 bits changed 46.25%	98 bits changed 55.68%

Table. 2: Bit Difference Comparison between existing SHA-160 SHA-176 algorithm after changing a single character

Figure 3 shows the graphical representation of table. 1, with execution time of existing SHA-160 and SHA-176 algorithm. According to the graph, there is a tendency that execution time for SHA-176 algorithm, increases with file size. But required time for the execution through SHA-176 is more than execution time for compared SHA-160. The time taken by SHA-176 algorithm is more as the message digest length is greater than that generated by SHA-160.

Figure 4 shows the graphical representation of table 6.2, the graph clearly shows that SHA-176 produces a greater AVALANCHE EFFECT as compared to SHA-160, i.e. a single change in the input produces a great change in the output. A change of a single letter in the input text produced a change of 98 bits in the output hash digest whereas in SHA-160 only 74 bits change.

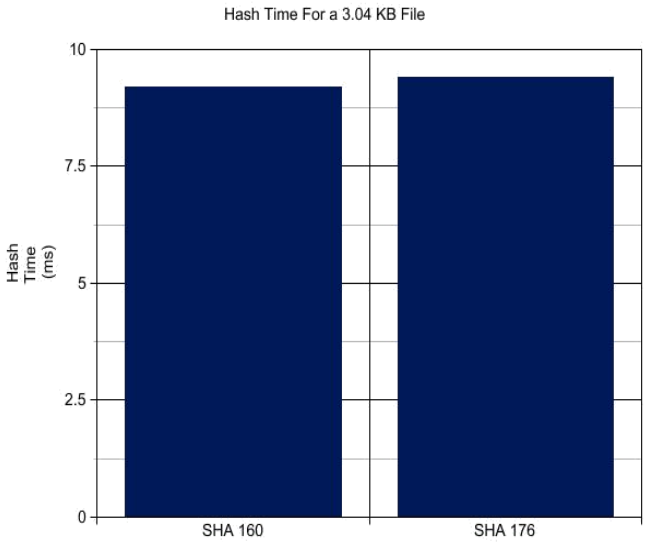


Fig. 3: Execution time (in mili Second) comparison between SHA-160 and SHA-176 for a 3.04 KB file

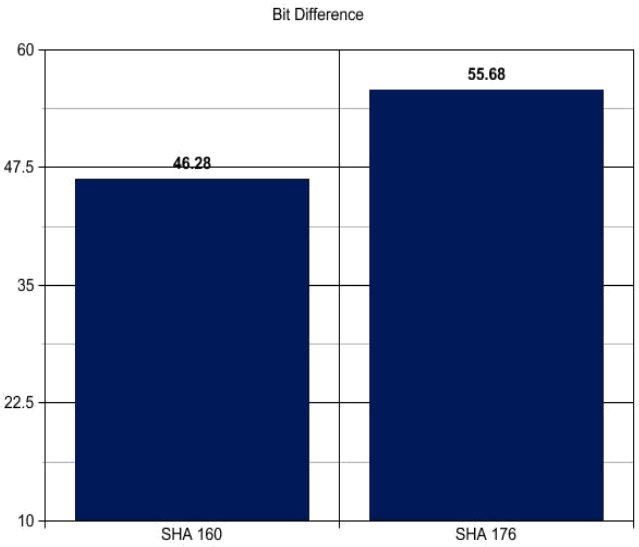


Fig. 4: Bit Difference Comparison between existing SHA-160 and SHA-176 algorithm after changing a single character.

Figure 5 shows the timing comparison between the outputs produced when hashing is done using SHA160 and SHA176 in TLS1.0.

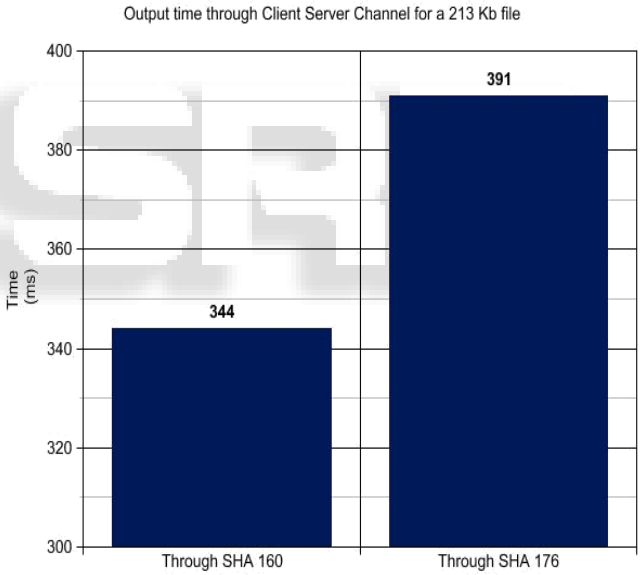


Fig. 5: Comparison of total time taken to encrypt a text and perform authentication in TLS1.0 using SHA-160 and SHA-176.

VIII. CONCLUSION

At the end we have come to a conclusion that the use of the improved authentication algorithm in TLS 1.0 will increase the security and integrity of the message during transmission of data from sender to receiver end. The SHA 176 model also requires less security by the transmission system during client server transmission as the improved algorithm is more secure and is more difficult to break. If the security and efficiency are the primary concern in TLS 1.0 authentication then one can use SHA-176 instead of existing SHA-160. The results show that SHA-176 algorithm is more powerful and secure than SHA-160 and cannot be easily broken. In future work can be done to improve the speed of the

algorithms used and try and achieve speed that can be used in 3G and 4G networks.

ACKNOWLEDGMENT

We are greatly indebted to Prof. Vikas Kaul, our internal guide for his constant support, guidance, co-ordination and encouragement. We also thank him for his moral support, patience, helpful suggestions and numerous discussions during the course of the project. His expertise in the subject of Data Security was a great boost to our efforts. We extend our sincere thanks to our Head of Department, Dr. Vinayak Bharadi. We would also like to thank our reputed principal Dr. B. K. Mishra, our project coordinators, all the staff members and lab assistants who have helped us in our project. We are grateful to our family, friends and classmates for their unconditional support and sincere feedback about out project. Our sincere thanks to the management of Thakur College Of Engineering And Technology for providing us with a platform and necessary facilities for accomplishing our project.

REFERENCES

[1] "Evolution of SHA 176 Algorithm", by Piyush Garg and Namita Tiwari in 2012 at IOSR Journal.
[2] "An approach for high security by using efficient SHA-176", by Snehlata Singh and Prof. Gajendra Singh Chandel in 2013 at IJRIT International Journal.
[3] A new Hash Function Based on Combination of Existing Digest Algorithms pub 2007.
[4] X. Wang, H. Yu and Y.L. Yin," Efficient Collision Search Attacks on SHA-0",(Pub 2005) .
[5] R. Anderson, E. Biham, and L. Knudsen, Serpent: A Proposal for the Advanced Encryption Standard, AES algorithm submission, June 1998
[6] "New Modified 256-bit MD5 algorithm with SHA Compression Function" International Journal of Computer Applications(0975-8887)
[7] William Stallings, "Cryptography and Network Security", 4th Edition.
[8] Behrouz A. Forouzan, "Cryptography and Network Security",5th Edition.
[9] Stephen Chapman, "MATLAB Programming for Engineers",2004.
[10] P. P Charles & P.L Shari, "Security in Computing: 4th edition", Prentice-Hall, lnc.2008.
[11] Atul Kahate ,Cryptography and Network Security, 2 EDITION
[12] http://en.wikipedia.org/wiki/RSA_(cryptosystem)
[13] http://en.wikipedia.org/wiki/Transport_Layer_Security
[14] http://en.wikipedia.org/wiki/Avalanche_effect.
[15] http://www.unixwiz.net/techtips/iguide-crypto-hashes.html
[16] https://www.coursera.org/course/crypto-preview/class/index
[17] www.cipherbyritter.com/LEARNING.HTM#Keyspace
[18] http://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html
[19] http://docs.oracle.com/javase/7/docs/api/javax/crypto/Cipher.html
[20] http://javadigest.wordpress.com/2012/08/26/rsa-encryption-example/
[21] http://docs.oracle.com/javase/7/docs/api/java/security/interfaces/package-summary.html
[22] http://docs.oracle.com/cd/B28196_01/idmanage.1014/b28171/oracle/security/crypto/core/RSA.html

