Journal of Engineering Science and Technology
Vol. 13, No. 3 (2018) 778 - 789

AN EFFICIENT HASH ALGORITHM TO PRESERVE DATA INTEGRITY

GURPREET K. SODHI, GURJOT S. GABA*

School of Electronics & Communication Engineering,
Lovely Professional University, Jalandhar, India - 144411
*Corresponding Author: er.gurjotgaba@gmail.com

Abstract

Recent advancements in the field of electronic commerce and internet banking have led to the growing need for a secure communication system. Various other areas including military require a highly reliable system so as to make sure that the shared data is confidential and unaltered.A negligence over these factors can lead to a huge and immutable loss. In this paper, a novel and Efficient Hash Algorithm(EHA)is presented which inherits the basic architecture of SHA-160. The performance of the proposed algorithm is evaluated by comparing it with the existing techniques which include MD2, MD5, SHA-160, SHA-256, SHA-384 and SHA-512. The comparison is done on the basis of NIST statistical test suite for random numbers and avalanche criteria. The results reveal that the suggested technique is more efficient in terms of randomness and throughput, thus, it can be efficiently used in any data sensitive environments.

Keywords: Hash, Data integrity, Message digest, Message authentication code, Security.

1. Introduction

The term communication is no longer confined to sharing of data between two of more parties, rather it now focuses on the data being shared securely. Securely here refers to retaining the confidentiality, integrity and authenticity of the shared data. Generally, a communication system may comprise of some critical information which is not to be disclosed to any unauthorized party, in order to prevent that information from being misused [1].

Now-a-days, the need of data assurance has increased to a higher extent due to the sensitivity of data and its usage in critical applications, where there is no com-

Nomenclatures

f	Function
K	Constant derived from S_{in} function
W	Word

Abbreviations

CDF	Check Determinant Factor
DFT	Discrete Fourier Transform
FIPS	Federal Information Processing Standard
MAC	Message Authentication Code
MD	Message Digest
NIST	National Institute of Standards and Technology
NSA	National Security Agency
SHA	Secure Hash Algorithm

promise with the data security [2]. Various schemes have been developed to verify that the received data is unaltered [1]. Also different security measures have been suggested in the literature [2] involving measures to retain confidentiality, integrity, authenticity and availability. Authenticity is of two types: Source Authentication & Message Authentication where the former assures the identity of the user and the latter assures the reception of unaltered data. Hayouni et al. [3], proposed various encryption and decryption algorithms to ensure confidentiality and authenticity.

Data integrity being of prime concern can be achieved using numerous hashing algorithms available. A Hash function maps an input of arbitrary length to a fixed output by using a noninvertible compression function. Hash functions have a property of being hard to reverse, which makes them efficient enough to be used for providing security services such as data integrity. They are widely used in applications such as emailing, digital signatures, electronic voting, e-commerce and online transactions. As compared to the cryptographic primitives such as symmetric and asymmetric ciphers, hash functions are better, in terms of efficiency [4].

Al-Mashhadi et al. [5] introduced changes in the existing Hash Algorithms in order to enhance the strength of security. There are basically two existing families used for calculating Hash: One-way Hash such as MD family, which constitutes of MD2, MD4, and MD5 [2] and SHA family, which constitutes of SHA-160, SHA-256, SHA-384, and SHA-512 [5]. The advancement here is the use of key for the purpose of producing the message digest. They are referred to as Message Authentication Code (MAC) [6] and Digital Signatures [5]. Message Digest produced by a hash algorithm is to be appended with original message, which is then compared with the value obtained at the receiver end; if both the values are same then data integrity is assured [5]. Hashing algorithms are considered to be highly secure, as for a given algorithm it is computationally impossible to find a message which is accurately same as the given message digest, or to find two distinct messages having same message digest. Anymodification in the message will result in a variation in the message digest [7]. Ghaeb et al. [8] used CDF (Check Determinant Factor) to measure data integrity. It involves appending of Determinant Factor for each data matrix before storing or transmitting the series of data. Since a communication system suffers from various types of attacks such as masquerade, disclosure, traffic analysis, content modification, sequence modification, source repudiation and destination repudiation etc. [2], therefore in order to achieve secure data transmission, there must be a system that provides authenticity of the source and message, so as to prevent the insertion of false data by an advisory. Various approaches have been presented in the past to maintain data integrity in a wireless network [5].

A detailed study on various Hash algorithms clearly shows a strategy followed by the researchers, which aims at increasing the message digest length in order to enhance the security and efficiency of a hash algorithm. While the two desired aims are achieved, an issue of increased bandwidth utilization and low throughput becomes an area of concern [3]. The proposed scheme since utilizes SHA-160 architecture, provides an efficient way to enhance security by increasing the complexity of the algorithm while retaining the message digest of smaller length.

2. Proposed Algorithm

The proposed algorithm is a secure one-way hashing algorithm of 160-bit, framed by enhancing the complexity of the existing hash function. It is clearly observed from Fig. 1 that EHA inherits the basic architecture of SHA-160 algorithm. SHA-160 algorithm is a cryptographic hash standard which was recommended by Network Working Group under RFC 1321. This was designed by NSA (National Security Agency) to be a part of digital signature algorithm. It was published by National Institute of Standards and Technology (NIST), as a U.S.A Federal information processing standard (FIPS). It takes an input of varying length and produces a 160-bit message digest. The whole compression function consists of 80 rounds [9].

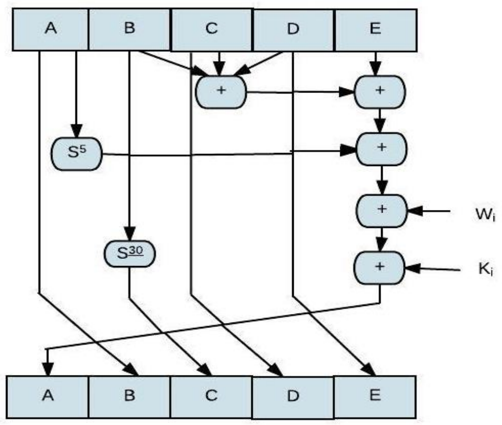

Fig. 1. SHA-160 compression function [10].

To enhance the impact of SHA-160 in terms of randomness, Expansion and S-box operations have been added into the existing architecture. These amendments increase the system complexity and thus help in preserving data integrity. The simulations of the proposed technique are carried out in MATLAB. The proposed scheme also enhances the system throughput by providing a message digest of smaller length as compared to other hashing algorithms of SHA family which includeSHA-256, SHA-384 and SHA-512. The whole process of the proposed algorithm is divided into three steps:

2.1. Pre-processing step

It constitutes of three steps: Padding the message, formation of blocks, and initialization of hash values.

Step 1: Hash algorithms have a constraint of input length. Therefore, padding has to be done in order to ensure that the padded message is multiple of 512. Padding here is done by appending single '1' bit followed by '0' bits till the length of bits in the message becomes congruent to 448 modulo 512.

Step 2: The message is to be divided into blocks of 512 bits each and these 512 bit blocks are further divided into 16 blocks of 32 bits each.

Step 3: Before further processing, five 32 bit words registers A, B, C, D and E have to be initialized with the values mentioned in Table 1.

Step 4: Save these above values in different variables like: $Ao = A\ , Bo = B$, $Co = C,\ Do = D,\ Eo = E$

Table 1.Buffer Values for SHA-160.

Registers	32-bit Words (Hexadecimal)
A	67452301
B	EFCDAB89
C	98BADCFE
D	10325476
E	C3D2E1F0

Further a function 'f' is introduced into the existing algorithm given in Fig. 2 to increase complexity.

- Each 512 bit block is divided into sixteen blocks of 32 bit each and these blocks are then expanded to eighty 32 bit blocks by using various mixing and shifting operations as mentioned in Eq. (1) [11]

$for\ t = 17{:}80$

$$W(t) = W(t-3)XorW(t-8)XorW(t-14)XorW(t-16) \ll 1 \tag{1}$$

$$end$$

- Four rounds of 20 bit operations are applied on the message blocks & buffer.
- Computation of 'f_t' function, where f_t(b, c, d) is a different nonlinear function in each round as given in Eqs. (2) to (5) [11]

$$for\ i = 1\ to\ 20 \geq\ f(B,C,D) = (B\ and\ C)\ or\ (not(B) and\ D) \tag{2}$$

$$for\ i = 21\ to\ 40 \geq f(B,C,D) = B\ Xor\ C\ Xor\ D \tag{3}$$

$$for\ i = 41\ to\ 60 \geq f(B,C,D) = (B\ and\ C)\ or\ (B\ and\ D)\ or\ (C\ and\ ) \tag{4}$$

$$for\ i = 61\ to\ 80 \geq f(B,C,D) = B\ Xor\ C\ Xor\ D \tag{5}$$

- W_t is derived from message blocks.

- K_t is a constant value derived from the S_{in} function as shown in Eqs. (6) to (9) [11]:

$$for\ i = 1\ to\ 20\ \geq K = 5A827999 \quad (6)$$

$$for\ i = 21\ to\ 40 \geq K = 6ED9EBA1 \quad (7)$$

$$for\ i = 41\ to\ 60 \geq K = 8F1BBCDC \quad (8)$$

$$for\ i = 61\ to\ 80 \geq K = CA62C1D6 \quad (9)$$

- Then the following operations are performed as shown from Eqs. (10) to (15) [11]

$$Temp = E + f(b,c,d) + (A \ll 5) + Wt + Kt \quad (10)$$

$$E = D \quad (11)$$

$$D = C\ (12) C = (B \ll 30) \quad (13)$$

$$B = A \quad (14)$$

$$A = Temp \quad (15)$$

- Here '+' indicates 2^{32} modulo addition and '<<'indicate left shift of the bits.

2.2. Output transformation step

The word registers are updated after execution of 2^{32} modulo addition operation between the initial values with final output value of word register as shown in Eqs. (16) to (20). After the preliminary message digest is generated, next 512 bit block of message and updated values of all the four words registers acts as the next input for the compression function. The processing of the last block of the input message leads to the generation of the message digest of the complete message.

$$A = mod((\boldsymbol{A_0} + A), 4294967296) \quad (16)$$

$$B = mod\big((\boldsymbol{B_0} + B), 4294967296\big) \quad (17)$$

$$C = mod\big((\boldsymbol{C_0} + C), 4294967296\big) \quad (18)$$

$$D = mod\big((\boldsymbol{D_0} + D), 4294967296\big) \quad (19)$$

$$E = mod((\boldsymbol{E_0} + E), 4294967296) \quad (20)$$

2.3. Introduction of '*f*' function

The proposed technique derives its strength from the '*f*' function, which has been integrated into the existing procedure. This '*f*' function constitutes of an expansion technique and Substitution block as shown in Fig. 2. The expansion technique used here is Symmetric extension which transforms the 32 bit input data into 48 bit data. Later, 2^{48} modulo addition is applied followed by the substitution box, which is used to convert the 48 bit data to 32 bit data. The details are given in *Appendix A*.

'*f*' function performs the following tasks: The expanded value of *register E* and function 'f_t' containing 48 bit data is provided as an input to perform 2^{48} modulo operation as shown in Eq. (21) and then the output that consists of 48 bit is given to the 'S' block as in Eq.(22) in order to convert the 48 bit data to 32 bits.

The left shifting of the bits is represented in the structure by 'S^n', where n is the number of bits to be shifted.

$$Y = mod\left(\left(Exp(E) + Exp(f_t)\right), 281474976710656\right) \tag{21}$$

$$Y1 = S(Y) \tag{22}$$

- Further, this '$Y1$' value consisting of 32 bits is used for proceeding operations.

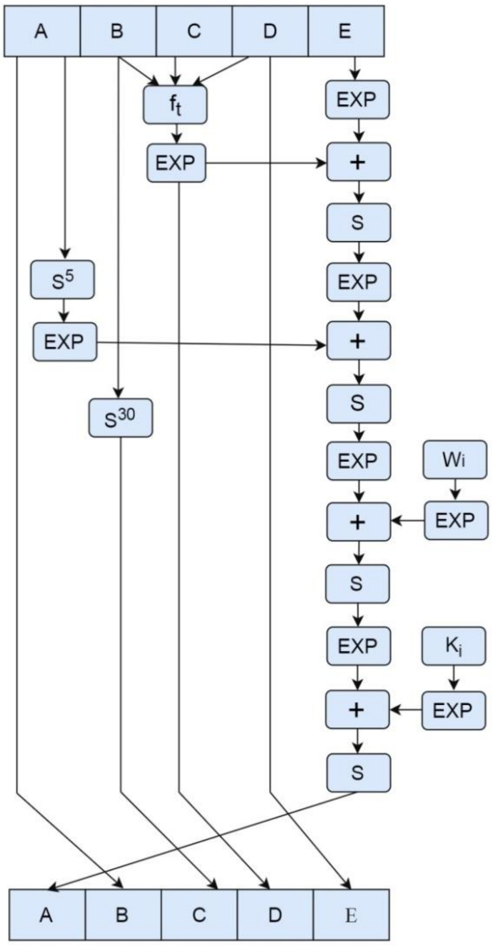

Fig. 2. Structure of EHA.

3. Results and Discussions

EHA has been evaluated on the basis of NIST tests of randomness and avalanche criteria. This is done using three different input values having different lengths. The length of the key has been normalized according to the key length of other schemes accordingly, in order to have proper comparison. So that unbiased output is obtained on a normalized platform. The inputs along with their corresponding hash values, resulting from various existing techniques and the recommended technique are given in Table 2.

Table 2. Message digest values for different inputs.

Techniques	Hash values		
	P	**World**	**Current work**
MD2	3687e026b0a5f81a 9fabd5205d804615	bb37795d20176658 552b6da07694861c	9c10bdde1d3aabbe 95038e9062f9bb44
MD5	83878c911713389 02e0fe0fb97a8c47a	7d793037a0760186 574b0282f2f435e7	fa2c20167700f35b9 8fa7ec01b11d84c
SHA-160	516b9783fca517ee cbd1d064da2d1653 10b19759	7c211433f02071597 741e6ff5a8ea34789 abbf43	eedb824ffed03619e 62b42f61f6e7ded24 010aae
SHA-256	148de9c5a7a44d19 e56cd9ae1a554bf6 7847afb0c58f6e12f a29ac7ddfca9940	486ea46224d1bb4fb 680f34f7c9ad96a8f 24ec88be73ea8e5a6 c65260e9cb8a7	fd878264b337d653 f712a333cdc8cb0ab 0bb08a9a5eb585c6 c6fc55b7acd27c8
SHA-384	049e7caf67d83409 ea363e89c09d67c7 f1fd1bd679016ad9 f422830ef105435e 12a4c2dcad5a9e5a 9602924d479574dc	ed7ced8487577360 3af90402e42c65f3b 48a5e77f84adc7a19 e8f3e8d310101022f 552aec70e9e1087b2 25930c1d260a	805dba18a4c056b8 391685ba8113a837 2d4f2bb217b2f83a6 1d2a34efb9f7b8b69 79e4ab51561a2516 8e915efcc11252
SHA-512	929872838cb9cfe6 578e11f0a323438a ee5ae7f61d41412d 62db72b25dac5201 9de2d6a355eb2d03 3336fb70e73f0ec0 afeca3ef36dd8a90d 83f998fee23b78d	11853df40f4b2b919 d3815f64792e58d08 663767a494bcbb38 c0b2389d9140bbb1 70281b4a847be775 7bde12c9cd0054ce3 652d0ad3a1a0c92ba bb69798246ee	f1b814483d475967 960077ad0868aacc 96f635c9c535b79f1 ed5e0e00f7afef710 830c73f5d06cb685 0093129433061bf8 7d02753e5f12f082d a4477dba54311
EHA	4283483b3e3a8b6d 5dcd699c274d1d37 74e17d44	617d79dc7a2b5669 1f5be1fa94a76db83 54d62f3	db13fea29dc8d457 25e6fe179dac64111 107a091

Further, NIST tests have been performed to evaluate the randomness of the message digests for the different set of inputs. These tests are used to calculate the P value for a binary sequence, which has to be greater than 0.01 for a sequence to be random [10].

A brief overview of the various NIST tests is given as:

i) Frequency test

Frequency test analyses the proportion of number of ones and zeros in the entire sequence. It checks the closeness between the number of ones and number of zeros. A sequence is said to be random if the proportion of both is close to each other [11]. The results in Table 3 illustrates that EHA produces better proximity between the count of ones and zeros as compared to the previously used techniques.

Table 3. NIST test results for frequency test.

Hash Technique	P-values		
	P	World	Current work
MD2	0.2888	0.7237	0.8597
MD5	0.3768	0.5959	0.4795
SHA-160	0.8744	0.8744	0.2059
SHA-256	0.4533	0.1498	0.4533
SHA-384	0.7595	0.2616	0.5403
SHA-512	0.2888	0.3768	0.5361
EHA	0.8744	0.1138	0.8746

ii) Binary derivative test

The Binary Derivative Test is performed using exclusive-or operation between successive bits until only one bit is left. Then, the ratio of number of ones to the length of entire sequence in each case is calculated. Finally, the average of the ratio of all the sequences is calculated, if the value lies near to 0.5, then the sequence is considered to be a random sequence [11]. The results in Table 4 indicate that the output of the proposed algorithm is random.

Table 4. NIST test results for binary derivative test.

Hash Technique	P-values		
	P	World	Current work
MD2	0.4849	0.5029	0.5049
MD5	0.5038	0.5042	0.4961
SHA-160		0.5110	0.5017
SHA-256	0.5006	0.5022	0.5014
SHA-384	0.5023	0.4997	0.5032
SHA-512	0.4980	0.4972	0.5021
EHA	0.4930	0.4974	0.5111

iii) Discrete Fourier transform test (DFT)

The purpose of DFT test is to find the peak heights in the Discrete Fourier Transform of a sequence. It detects the presence of similar patterns in the sequence which further indicates a divergence from the assumed randomness. The focus is to check if the number of peaks exceeding the 95% threshold is significantly different than 5%. The results for DFT test are summarized in Table 5.

iv) Approximate entropy test

The objective of this test is to calculate the frequency of all the overlapping bit patterns present in the sequence. It compares the frequency of overlapping blocks of two sequential lengths with the expected outcome for a random sequence. The results are given in Table 6.

Table 5. NIST test results for DFT test.

Hash Technique	P-values		
	P	World	Current work
MD2	0.5164	0.8711	0.5164
MD5	0.3304	0.5164	0.5164
SHA-160	0.1000	0.4682	0.1000
SHA-256	0.3588	-	0.7308
SHA-384	0.0027	0.4537	0.7787
SHA-512	0.7456	0.6265	0.1233
EHA	0.4682	0.4688	0.4680

Table 6. NIST test results for approximate entropy test.

Hash Technique	P-values		
	P	World	Current work
MD2	0.6169	0.3499	0.5739
MD5	0.4108	0.8312	0.6896
SHA-160	0.8897	0.8018	0.6803
SHA-256	0.9963	0.7720	0.9080
SHA-384	0.9954	0.9915	0.9712
SHA-512	0.9841	0.9801	0.9965
EHA	0.5619	0.9163	0.8896

v) Maurer's "Universal statistical" test

This test emphasizes on finding out if a sequence can be compressed without any loss of information. A sequence is said to be non random if it is significantly compressible [11]. The results are summarized in Table 7.

Table 7. NIST test results for Maurer test.

Hash Technique	Maurer test		
	P	World	Current work
MD2	0.8981	0.9880	0.9664
MD5	0.9753	0.9985	0.9675
SHA-160	0.9914	0.9135	0.9818
SHA-256	0.9976	0.9690	0.9935
SHA-384	0.9836	0.9978	0.9993
SHA-512	0.9831	0.9600	0.9850
EHA	0.9943	0.9892	0.9926

As clearly noticed from Tables 3 to 7, the proposed technique performs better bypassing the NIST criteria of generating a random message digest. Thus, signifies its effectiveness as a Hash algorithm.

A message digest is designed in order to protect the integrity of a piece of data and to considerably detect any kind of changes or alteration in any part of the message. Also, one message digest specifically represents particular data content and thus it can reference a change made intentionally or unintentionally. Thus, there must be a change in a particular message digest following a change in the data file. To study this parameter, a separate test has been applied to the hash values, i.e., Avalanche Test. This test is used to calculate the change in the output with respect to a change in the output. This is known as the avalanche effect and it's represented in the formula as given in Eq. (23) [1]

$$\text{Avalanche Effect} = \frac{\boldsymbol{No.of bits flipped}}{\boldsymbol{Total no.of bits in the seqence}} \times 100 \tag{23}$$

Higher the avalanche effect higher is the efficiency of the technique. This test has been applied by altering a single character of the input value, considering existing techniques and comparing the results with the presented one. The Avalanche Test results are summarized in Table 8.

Table 8. Avalanche test analysis.

Original Input	**Altered Input**	**Avalanche Effect (%)**						
		MD2	**MD5**	**SHA-160**	**SHA-256**	**SHA-384**	**SHA-512**	**EHA**
	G	50.00	46.87	46.45	52.43	51.82	49.80	46.90
World	Worlk	52.75	45.31	52.50	46,87	51.30	50.00	51.87
Current work	Current work	46.09	54.68	52.50	45.31	53.90	50.39	49.87

It can be clearly noticed that the recommended technique performs well under this criteria too, thus demonstrating its efficiency.

Further EHA is evaluated on another factor, i.e., throughput which is explained as the maximum output that can be produced by a system within the specified resources such as bandwidth. It is desirable to make the best use of resources by reducing the amount of redundant data transmission. Throughput can be measured by using Eq. (24) [2]

$$Throughput = \frac{Data without overheads}{Total data transmitted} \tag{24}$$

EHA produces an output of 160 bits given the input data of variable length. When compared to other hash algorithms of SHA family, EHA enhances the throughput of the system significantly.

4. Conclusion

This paper presents a novel technique which is a result of the modification of the basic architecture of SHA-160 by embedding in it the expansion and substitution function. The recommended technique has been tested on client server model using statistical test suite introduced by NIST and avalanche criteria. The conclusions drawn are listed as:

- The analysis of the different test results concludes that the proposed algorithm performs better than most of the existing techniques such as MD2, MD5, SHA-160, SHA-256 and SHA-384.

- This scheme serves its best purpose by generating a random message digest of 160 bit length, which is less than the outputs generated by other schemes of SHA family and thus the proposed scheme enhances the overall system throughput.
- The proposed hash can also be effectively used along with the incorporation of a security key for the implementation of MAC, to provide data as well as source authentication.
- Hence, the presented hash algorithm can be proficiently applied in an environment demanding data security.

References

1. Malinowski, C.; and Noble, R. (2007). Hashing and data integrity reliability of hashing and granularity size reduction. *Digital Investigation*, 4(2), 98-104.
2. Stallings, W. (2014). *Cryptography and network security: Principles and practices*. New York, NY: Pearson Education.
3. Hayouni, H.; Hamdi, M.; and Kim, T.-H. (2015). A novel efficient approach for protecting integrity of data aggregation in wireless sensor networks. *Proceedings* of 2015 *International Wireless Communications and Mobile Computing Conference*, 1193-1198.
4. Al-Riyami, A.; Zhang, N.; and Keane, J. (2016). Impact of hash value truncation on ID anonymity in WSN. *Ad Hoc Networks*, 45, 80-103.
5. Al-Mashhadi, H.M.; Abdul-Wahab, H.B.; and Hassan, R.F. (2014). Secure and time efficient hash-based message authentication algorithm for wireless sensor networks. Proceedings of 2014 *Global Summit on Computer & Information Technology* (*GSCIT*), 1-6.
6. Chang, C.-Y.; Chang, C.-T.; Zhao, L.; Ding, Z.; and Chen, C.-C. (2017). A stepwise multichannel MAC protocol for improving bandwidth utilization in wireless Ad Hoc networks. Proceedings of the *IEEE* Systems Journal, 11(4), 2444-2455.
7. Ravilla, R.; and Shekar, C. (2015). Enhancing the security of MANETs using hash algorithm. *Proceedings of Elsevier IMCIP*., 196-206.
8. Ghaeb, J.A.; Smadi, M.A.; and Chebil, J. (2011). A high performance data integrity assurance based on determinant technique. *Future Generation* Computer Systems, 27(5), 614-619.
9. Chabaud, F.; and Joux, A. (1998). *Differential collisions in SHA-0. Advances in cryptology, Crypto'98. Springer-Verlag,* 56-81.
10. Eastlake D.; and Hansen T. (2006). US secure hash algorithms (SHA and HMAC-SHA). RFC, Network Working Group.

Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; Dray, J.; and Vo, S. (2010). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. Special Publication 800-22, Revision 1a. National Institute of Standards and Technology

Appendix A

S-boxes used in the proposed technique

In the presented work, eight Substitution boxes have been used as a part of the '*f*' function, which is applied in the existing SHA-160 Algorithm. The substitution boxes are used in order to convert the expanded 48 bit data into 32 bit data [2].

The eight S-boxes used are shown in Fig. A-1.

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Fig. A-1. S-boxes [2].