International Journal of Theoretical & Applied Sciences, 11(2): 41-47(2019)

ISSN No. (Print): 0975-1718
ISSN No. (Online): 2249-3247

Design and Mathematical Structure of Cryptographic Hash Function SHA-512

Arun Kumar Sharma
Department of Computer Science & Engineering,
NIT Hamirpur, India

(Corresponding author: Arun Kumar Sharma)
(Received 30 September, 2019, accepted 19 November, 2019)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Cryptography is an essential part of information and communication technology. Security of information is provided with cryptography. Cryptography forms the basis in Security of electronic commerce, computer passwords and ATM cards. In cryptography, strong security is ensured by strong algorithm of mathematics hence assuring safety of the data to wide range of users. In a properly designed cryptographic scheme, the security of the scheme is dependent on the algorithm used. In the present paper, we discuss the design and its mathematical processing of data at various stages of SHA family in Cryptographic Hash functions particularly SHA-512. Further, we illustrate the mathematical processing of SHA-512.

Keywords: Cryptography, Hash Functions, E-commerce, Secure Hash Algorithm, Security, SHA-512.

I. INTRODUCTION

We are living in the information age. The information storage and exchange become electronic. The channels through which the commutation of data takes place is not secure. The information being transmitted is vulnerable to various attacks. Anyone can easily read the data that commutes over the channels which are not safe. Therefore, information security has become most challenging aspects of communication. Cryptography helps us to secure our data from hackers. Cryptography changes our data from original form to the form that is not easily recognizable and that data commutes over channels which are not safe and if someone hacks our data then he cannot understand what is written. The origin of the word Cryptography find its mention in the Greek words: Kryptos and Graphein, Kryptos meaning "hidden" or "secret" and Graphein meaning "writing". Hence, the cryptography is defined as the art and science of secret writing or hidden writing. The techniques used in cryptography changes the original message at sender end to unreadable form and transmit this over insecure channels. At the receiver end again data changes to original message with the help of cryptographic techniques [2, 5, 10, 14, 15, 16]. Hence, our data securely travels over insecure channels. The first design of cryptographic hash functions appeared in late 1970s. A wide range of proposals appeared in 1980s. During the 1990s, there was a great growth in the number of hash functions in a very brief period of time, but most of the proposals had security flaws which were identical. MD5 and SHA-1 were used in a large number of applications, resulting in 000000000000000 name "Swiss army knife" of cryptography. Undermining the importance of hash functions, very few efforts were spent to study the formal definitions and foundations corresponding to Hash Functions. In 2004, Wang et al. efforts made Cryptanalysis reach a point where finding collision for MD5 became very easy.

II. RELATED WORK

For SHA-1, a considerable reduction corresponding to the security margin was achieved. The breakthrough has resulted in a wide range of research, further corresponding to foundational research and new construction [12, 13, 17].

A. SHA-1

The development of SHA-1 was done keeping in mind the Capstone project of the U.S. Government. The original specification is now mainly known as SHA-0 of the algorithm which was published in 1993 under the heading Secure Hash Standard by the U.S. Government Standard agency NIST (National Institute of Standard and Technology). It was revoked by NSA shortly after its publication and replaced by a revised version published in 1995 commonly known as SHA-1. Collisions against the full SHA-1 algorithm can be generated using shattered attacks and further resulting in broken hash function. SHA-1 produces a hash digest of 160 bits(20bytes).

B. SHA-2

Secure Hash Algorithm-2 (SHA-2) is a unique collection of cryptographic hash functions designed by the National Security Agency (NSA) of the United States, that was firstly published in 2001. Merkle-Damgard structure forms the basis of its construction. SHA-2 mainly contains two Hash Algorithms: SHA-256 and SHA-512. SHA-224 is a variant of SHA-256with different starting values and truncated output.

SHA-384, the lesser known SHA-512/224 and SHA-512/256 all form the variation of SHA-512. SHA-512 is much safer than SHA-256 and is usually faster than SHA256. The output in bits is given by the extension to the "SHA" name.

SHA-224-Output 224 bits (28bytes).
SHA-256-Output 256 bits (32bytes).
SHA-384-Output 384 bits (48bytes).
SHA-512-Output 512 bits (64bytes).

C. SHA-3

The development of Secure Hash Algorithm-3 (SHA-3) took place on August 5, 2015 by NIST. SHA-3 is a subset of the crypto-graphic primitive family called keccak. The keccak algorithm is the work of Guido Bertoni, Joan Daemen, Michael Peeters and Gills Van Assche. Sponge Construction lays the basis of keccak which can also be used to create other cryptographic primitive similar to stream ciphers. SHA-3 offers same output size as SHA-2 that is of 224,256,384 and 512 bits, see [4, 9].

Algorithm	Size of Message Digest	Message Block Size	Collision
SHA-0	160	512	Yes
SHA-1	160	512	Yes
SHA-256/224	256/224	512	No
SHA-512/324	512/324	1024	No

SHA-512 is one of the most secure hash functions available today, see [1, 3, 6]. Operation is much faster than any other member of SHA-Family. It forms the latest version, has more complex structure than, and the corresponding message digest is longest. Though there are quite a few types of attacks on SHA, none of them are completely successful.

Actually, it is not so easy to decrypt the output from a hash function. There are different types of attacks employed to decrypt SHA-512. Following are the most famous one.

1) Preimage Attack:
   It defines a message that has a specific Hash value.
2) Collision Attack:
   Birthday Attack, is an example of collision attack. It takes o($2^{n/2}$) times where, n is length of the output of hash function SHA-512. Assume that for 32 byte input, the time taken by machine is 0.22s(2^{-2}s) for 65536(= 2^{16}) computations. So, 2^{256} computations would be completed in $2^{240}.2^{16}$ computations which would take

$$2^{240} * 2^{-2} = 2^{238}\ 10^{72}s\ 3.17 * 10^{64} Years.$$

3) Second Preimage Attack:
   SHA-1 which employ 256 bits is considered to be broken, since a collision was identified at 2^{69} operations much less than 2^{80}. None of the above attack can crack a hash generated by SHA-2 algorithms with the best of the hardware available on earth. So, cheers as SHA-512 is still secure and will be.

III. DESIGN OF SHA-512

The SHA-512 algorithm uses one- way hash function created by United States National Security Agency (NSA).

A one-way hash function also called as message summary or compression function denotes a mathematical function that takes the entering variable length and change it to a binary sequence of fixed length. The one-way hash function is designated in such a manner that it is difficult to change the order of the process. The hash function is good if it is difficult to find two strings that will produce similar hash value.

A message which consists maximum length of bits is taken as input by the algorithm and an output comprising of a 512-bit message digest is produced. The processing of input is done in 1024- bits blocks. The steps used in algorithm are as follows [7, 8, 11, 14].

Algorithm of SHA-512

1) Step-1: Append the bits with padding

Let us take the message "M". Convert the characters of the message into ASCII codes. Convert the characters of the message from ASCII codes to binary. The original message to be hashed is padded with binary digits of 1 and 0's so that its length becomes congruent to 896 modulo 1024. The padding is usually followed by many 0's.

2) Step-2: Append Length Field

The appending of the block of 128-bits is done to the message. This block is treated as an unsigned 128-bit integer and contains the length of the original message (before the padding).

3) Step-3: Initialize Hash Buffers

The intermediate and final results corresponding to the hash function are held by 512-bit buffer. The buffer can be represented as eight 64-bit registers.

4) Step-4: Words and its expansion

SHA-512 operates on words; its orientation or inclination is towards word. A word contains 64-bits. This further states that after addition of padding and length field is done to original message each blocks of message comprises sixteen 64-bit words. 16 words, each of 64-bits=512- bits.

5) Step-5: Process message in 1024-bit blocks

The heart of the algorithm is its compression function.80 constants are used one in each step. 80 words are also used one in each step. All these are mixed together and then a new set of eight buffers is created. The buffers created after first round becomes the initial buffers for second round. And the buffers created after second round becomes the initial buffers for third round and so on. At the end of processing the addition of initial buffers is done with values created from step 79. The last operation is called the final adding.

6) Step-6: Final Adding

After processing the data 80 times (80 rounds) as mentioned above, we get the final output as below:

$A = a_0+a_{79}$

$B = b_0+b_{79}$

$C = c_0+c_{79}$

$D = d_0+d_{79}$

$E = e_0+e_{79}$

$F = f_0+f_{79}$

$G = g_0+g_{79}$

$H = h_0+h_{79}$

Therefore, the last output of 512-bits of the hash function is "ABCDEFGH". It is known as message digest. Further, we change this binary message into hexadecimal form, which gives 128 hexadecimal characters.

IV. ILLUSTRATION

SHA-512 emphasizes that the initial message length should be less than 2^{128} -bits. A digest of 512-bits from a multiple-block message is created. The length of each block is 1024-bits. Now let us discuss the mechanism of SHA-512 algorithm.

1) Step-1

Let us take the message (M), "Cryptography" and convert the characters of the message into ASCII, then converting each character of the message from ASCII codes to binary.

Thus, the binary form of the message is
M=0100001101110010011110010111000001110100011011110110011101110010011000010111000001101000011 11001

The message (M) is of 96-bits. Therefore, append bit '1' in the extreme right and remaining zeros up to the 896-bit message. Therefore,
M=0100001101110010011110010111000001110100011011110110011101110010011000010111000001101000011110011000............800 times 0

2) Step-2

Since the original message (M) has the length of "96-bits", that is $M = 96_{10}$.
Then, its binary representation is
$96_{10} = 01100000_2$. This is short of 128-bits. Therefore, append zeros extreme left to make 128-bits. Thus, 128-bit string is as follows:
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
0000000000000000000000000000000001100000
Now, we append this string with the message string obtained above and we get the message string of 1024-bits.

3) Step-3

512-bit buffers are used to hold intermediate and final result of the hash function. The buffers can be represented as eight 64-bit registers (a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0). These registers are initialized to the following 64-bit integers (hexadecimal values).
a_0 = 6A09E667F3BCC908
b_0 = BB67AE8584CAA73B
c_0 = 3C6EF372FE94F82B
d_0 = A54FF53A5F1D36F1
e_0 = 510E527FADE682D1
f_0 = 9B05688C2B3E6C1F
g_0 = 1F83D9ABFB41BD6B
h_0 = 5BE0CD19137E2179

4) Step-4

Now, dividing 1024-bits message into 16 words each of 64-bits starting from w_0 to w_{15}as
w_0=01000011011100100111100101110000011101000 1101111011001110111001 0
w_1=01100001011100000110100001111001100000000 0000000000000000000000 and so on.

Now, next 64 words are calculated from previous generated words as

$$w_t = w_{t-16} + \sigma_0^{512}(w_{t-15} + w_{t-7} + \sigma_1^{512}(w_{t-2}))$$

5) Step-5

Now we proceed towards compression function.

Round-1

We have discussed the initial buffers in Step-3. Let us convert the values of buffers from Hexadecimal to binary starting from a_0 to h_0 where

$$a_0 = \begin{pmatrix} 0110101000001001111001100110011111110011 \\ 101111001100100100001000 \end{pmatrix}$$

$$b_0 = \begin{pmatrix} 1011101101100111101011101000010110000100 \\ 110010101010011100111011 \end{pmatrix}$$

and so on.

Now,

$$T_1 = h_0 + Ch(e_0, f_0, g_0) + \sum_1^{512}(e_0) + w_0 + K_0$$

where,

$$Ch(e_0, f_0, g_0) = (e_0 \wedge f_0) \oplus (-e_0 \wedge g_0)$$

that is,

$$Ch(e_0,f_0,g_0) = \begin{pmatrix} 0001111110000010111001001100011000111101 1 \\ 0010011100111101001110 11 \end{pmatrix}$$

and

$$\sum_{1}^{512}(e_0) = ROTR^{14}(e_0) \oplus ROTR^{18}(e_0) \oplus ROTR^{41}(e_0)$$

Now, K_0=428A2F98D728AE22

$$K_0 = \begin{pmatrix} 0100001010001010001011111001100011010111 \\ 001010001010111000100010 \end{pmatrix}$$

$$w_0 = \begin{pmatrix} 0100001101110010011110010110000001110100 \\ 011011110110011101110010 \end{pmatrix}$$

Therefore,

$$T_1 = \begin{pmatrix} 1110111001000101010111101010100110000001 \\ 001010001111000100101110 \end{pmatrix}$$

Now,

$$T_2 = \sum_{0}^{512}(a_0) + Maj(a_0 + b_0 + c_0)$$

where,

$$\sum_{0}^{512}(a_0) = (e_0)ROTR^{28}(a_0) \oplus ROTR^{34}(a_0) \oplus ROTR^{39}(a_0)$$

$$\sum_{0}^{512}(a_0) = \begin{pmatrix} 1010000001000110001001101101101010110101 0 \\ 101011001000000011000010 \end{pmatrix}$$

And

$$Maj(a_0,b_0,c_0) = (a_0 \wedge b_0) \oplus (a_0 \wedge c_0) \oplus (b_0 \wedge c_0)$$

$$Maj(a_0,b_0,c_0) = \begin{pmatrix} 0011101001101111111001100110011111100110 \\ 100000001010000000101000 \end{pmatrix}$$

Therefore,

$$T_2 = \begin{pmatrix} 1101101011111110000110100000011101010100 01 \\ 011011101100010111101010 \end{pmatrix}$$

Now,

$$a_1 = T_1 + T_2$$

that is,

$$a_1 = \begin{pmatrix} 1100100101000001100100101100011011010010 \\ 100101111100101100011000 \end{pmatrix}$$

$$b_1 = \begin{pmatrix} 0110101000000100111100110011001111111001 1 \\ 101111001100100100001000 \end{pmatrix}$$

$$c_1 = \begin{pmatrix} 1011101101100111101011101000010110000100 \\ 110010101010011100111011 \end{pmatrix}$$

$$d_1 = \begin{pmatrix} 0011110001101110111100110111001011101111 \\ 100101001111100000101000 \end{pmatrix}$$

$$e_1 = \begin{pmatrix} 1001001110010101010000111110001111100000 \\ 010001100010100000011111 \end{pmatrix}$$

$$f_1 = \begin{pmatrix} 0101000100001110010100100111111110101101 \\ 111001101000001011010001 \end{pmatrix}$$

$$g_1 = \begin{pmatrix} 1001101100000101011010001000110000101011 \\ 001111100110110000011111 \end{pmatrix}$$

$$h_1 = \begin{pmatrix} 0001111110000011110110011010101111111011 \\ 010000011011110101101011 \end{pmatrix}$$

Round-2

Buffers generated in Round-1 are initial value buffers for Round-2. Let us calculate round functions to calculate buffers of Round-2.

Now,

$$T_1 = h_1 + Ch(e_1,f_1,g_1) + \sum_{n=1}^{512}(e_1) + w_1 + K_1$$

where,

$$Ch(e_1,f_1,g_1) = (e_1 \wedge f_1) \oplus (-e_1 \wedge g_1)$$

$$Ch(e_1,f_1,g_1) = \begin{pmatrix} 0001100100000010001101010011011111010101 1 \\ 011111100100010000010001 \end{pmatrix}$$

and,

$$\sum_{1}^{512}(e_1) = ROTR^{14}(e_1) \oplus ROTR^{18}(e_1) \oplus ROTR^{41}(e_1)$$

$$\sum_{1}^{512}(e_1) = \begin{pmatrix} 1000101110011000010110101001001101001011 \\ 011110001011000011000011 \end{pmatrix}$$

$$w_1 = \begin{pmatrix} 0110000101110000011010000111100110000000 \\ 000000000000000000000000 \end{pmatrix}$$

K_1=3956C25BF348B538

$$K_1 = \begin{pmatrix} 0011100101010110110000100101101111110011 \\ 010010001011010100111000 \end{pmatrix}$$

Therefore,

$$T_1 = \begin{pmatrix} 0000101011000110101111001000101010011110 \\ 111100000000011101110111 \end{pmatrix}$$

Now,

$$T_2 = \sum_{0}^{512}(a_0) + \mathrm{Maj}(a_1 + b_1 + c_1)$$

where,

$$\sum_{0}^{512}(a_1) = \mathrm{ROTR}^{28}(a_1) \oplus \mathrm{ROTR}^{34}(a_1) \oplus \mathrm{ROTR}^{39}(a_1)$$

$$\sum_{0}^{512}(a_1) = \begin{pmatrix} 0001000000111110001101100110111001001011 1 \\ 110110111100101111111001 \end{pmatrix}$$

and

$$\mathrm{Maj}(a_1, b_1, c_1) = \begin{pmatrix} 1110101101000001101001101100011111010010 \\ 100111101100101100011000 \end{pmatrix}$$

Therefore,

$$T_2 = \begin{pmatrix} 1111101110111111000010011101001000110101 0 \\ 011110101001011100010001 \end{pmatrix}$$

Now,

$$a_2 = T_1 + T_2$$

that is

$$a_2 = \begin{pmatrix} 0001000000111110001101100110111001001011 1 \\ 110110111100101111111001 \end{pmatrix}$$

$$b_2 = \begin{pmatrix} 1100100101000000110010010110001101101001 0 \\ 100101111100101100011000 \end{pmatrix}$$

$$c_2 = \begin{pmatrix} 0110101000000100111100110011001111111001 1 \\ 101111001100100100001000 \end{pmatrix}$$

$$d_2 = \begin{pmatrix} 1011101101100111101011101000010110000100 \\ 110010101010011100111011 \end{pmatrix}$$

and

$$e_2 = d_1 + T_1$$

$$e_2 = \begin{pmatrix} 0100011100110101101011111111110110001110 \\ 100001001111111110011111 \end{pmatrix}$$

$$f_2 = \begin{pmatrix} 1001001110010101010000111110001111100000 \\ 010001100010100000011111 \end{pmatrix}$$

$$g_2 = \begin{pmatrix} 0101000100001110010100100111111110101101 \\ 111001101000001011010001 \end{pmatrix}$$

$$h_2 = \begin{pmatrix} 1001101100000101011010001000110000101011 \\ 001111100110110000011111 \end{pmatrix}$$

Similarly, up to the round-79 the values of buffers after round-79 are as

$$a_{79} = \begin{pmatrix} 1101001110110001001110111011110011110001 \\ 010100111011111001111000 \end{pmatrix}$$

$$b_{79} = \begin{pmatrix} 0110100100000011000111000100100001100011 \\ 001001001111101111110101 \end{pmatrix}$$

$$c_{79} = \begin{pmatrix} 0000111110010011010010001101001110001010 \\ 101101100110001100100101 \end{pmatrix}$$

$$d_{79} = \begin{pmatrix} 1010011011101010110010100101110011110001 \\ 011011010111101000100001 \end{pmatrix}$$

$$e_{79} = \begin{pmatrix} 1011100010100111000100001101101011001111 \\ 100011110001011110101011 \end{pmatrix}$$

$$f_{79} = \begin{pmatrix} 0010000110100110111001011101110001010110 \\ 000000111001001010111100 \end{pmatrix}$$

$$g_{79} = \begin{pmatrix} 0111010110101100010101110010011101101001 \\ 100011010100000111101001 \end{pmatrix}$$

$$h_{79} = \begin{pmatrix} 0111110111010111001010100010111100000000 \\ 011001001010100010100111 \end{pmatrix}$$

$$A = a_0 + a_{79}$$

$$B = b_0 + b_{79}$$

$$C = c_0 + c_{79}$$

$$D = d_0 + d_{79}$$

$$E = e_0 + e_{79}$$

$$F = f_0 + f_{79}$$

$$G = g_0 + g_{79}$$

$$H = h_0 + h_{79}$$

Therefore,

$$A = \begin{pmatrix} 0001110110111011001000100010010011100101 \\ 000100001000101110000000 \end{pmatrix}$$

$$B = \begin{pmatrix} 0010010001101010110010101100110101100111 \\ 101011111010001100110000 \end{pmatrix}$$

$$C = \begin{pmatrix} 0100110000000001000110011010001100111101 0 \\ 010010100101101101001101 \end{pmatrix}$$

$$D = \begin{pmatrix} 0011110000111010101011111001011101010000 \\ 100010101011000100010010 \end{pmatrix}$$

$$E = \begin{pmatrix} 0000100010110101011000110101101001111101 \\ 011101011001101001111100 \end{pmatrix}$$

$$F = \begin{pmatrix} 1010110010101100010011100110100010000001 \\ 010000001110111101101101 1 \end{pmatrix}$$

$$G = \begin{pmatrix} 1001010100110000000110000111100110110010 0 \\ 110011101111111110101010 0 \end{pmatrix}$$

$$H = \begin{pmatrix} 1101100110110111111100110100100000001001 1 \\ 111000101100101000100000 \end{pmatrix}$$

$$ABCDEFGH = \begin{pmatrix} 0001110110111011001000100010010011100101 \\ 0001000010001011100000000010010001101010 \\ 1100101011001101011001111010111110100011 \\ 0011000000100110000000001000111100010001 10 \\ 0111101001001010010110110100110101001100 \\ 0011101010101111100101110101000010001010 \\ 1011000100010010000001000101101010110001 1 \\ 0101101001111101011101011001101001111100 \\ 1010110010101100010011100110100010000001 \\ 0100000011101111011011011100101010011000 0 \\ 0011000011110011011001001100111011111111 \\ 0101010011011001101101111111011101001000 \\ 00010011111000101100101000100000 \end{pmatrix}$$

Therefore, last output is of 512-bits. Converting binary Hexadecimal, we get

A=1dbb2224e5108b80

B=246acacd67afa330

C=4c023c467a4a5b4d

D=3c3aaf97508ab112

E=08b5635a7d759a7c

F=acac4e688141dedb

G=953030f364ceff54

H=d9b7f34813e2ca20

Therefore, message digest has 128 characters as

1dbb2224e5108b80246acacd67afa3304c023c467a4a5
b4d3c3aaf97508ab11208b5635a7d759a7cacac4e6881
41dedb953030f364cef54d9b7f34813e2ca20.

V. CONCLUSION

A cryptographic hash function is a process to produce a fixed size output of enciphered text from the text having variable size. We discussed the various cryptographic hash functions and particularly the design of SHA-512. Also, we explained the processing of flow of data in SHA-512 with the help of an illustration.

REFERENCES

[1]. Dobraunig, C., Eichlseder, M., & Mendel, F. (2015, November). Analysis of SHA-512/224 and SHA-512/256. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 612-630). Springer, Berlin, Heidelberg.

[2]. Forouzan, B. A., & Mukhopadhyay, D. (2015). *Cryptography and network security*. Mc Graw Hill Education (India) Private Limited.

[3]. Grembowski, T., Lien, R., Gaj, K., Nguyen, N., Bellows, P., Flidr, J., ... & Schott, B. (2002, September). Comparative analysis of the hardware implementations of hash functions SHA-1 and SHA-512. In *International Conference on Information Security* (pp. 75-89). Springer, Berlin, Heidelberg.

[4]. Handschuh, H., Knudsen, L. R., & Robshaw, M. J. (2001, April). Analysis of SHA-1 in encryption mode. In *Cryptographers' Track at the RSA Conference* (pp. 70-83). Springer, Berlin, Heidelberg.

[5]. Hłobaż, A. (2019, October). Statistical Analysis of Enhanced SDEx Encryption Method Based on SHA-256 Hash Function. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)* (pp. 238-241). IEEE.

[6]. Lee, S. H., & Shin, K. W. (2018, January). An efficient implementation of SHA processor including three hash algorithms (SHA-512, SHA-512/224, SHA-512/256). In *2018 International Conference on Electronics, Information, and Communication (ICEIC)* (pp. 1-4). IEEE.

[7]. Menezes Alfred, J., van Oorschot Paul, C., & Vanstone Scott, A. (1996). Handbook of applied cryptography.

[8]. Paar, C., &Pelzl, J. (2009). *Understanding cryptography: a textbook for students and practitioners.* Springer Science & Business Media.

[9]. Preneel, B., Govaerts, R., &Vandewalle, J. (1989). *Cryptographically secure hash functions: an overview.* In: ESAT Internal Report, KU Leuven.

[10]. Schneier, B., & Kelsey, J. (1998, January). Cryptographic support for secure logs on untrusted machines. In *USENIX Security Symposium* (Vol. **98**, pp. 53-62).

[11]. Seberry, J., & Pieprzyk, J. (1989). *Cryptography: an introduction to computer security*. Prentice-Hall, Inc.
[12]. Sharma, A. K. (2018). Content-Based Filtering in Movie Recommendation. *International Journal of Electrical, Electronics and Computer Engineering*, *7*(2): 106-109.
[13]. Sharma, A. K. (2019). Safety Application in Android. *International Journal on Emerging Technologies, 10*(1): 234-238.
[14]. Stallings, W. (2006). *Cryptography and network security, 4/E*. Pearson Education India.
[15]. Stinson, D. R., & Paterson, M. (2018). Cryptography: theory and practice. CRC press.
[16]. Sun, W., Guo, H., He, H., & Dai, Z. (2007, October). Design and optimized implementation of the SHA-2 (256, 384, 512) hash algorithms. In *2007 7th International Conference on ASIC* (pp. 858-861). IEEE.
[17]. Xiaoyun, W. (2005). Finding collisions in the full SHA-1. In *CRYPTO 2005* (pp. 17-36).