THINK INDIA JOURNAL

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

Improved Cryptographic Technique Using Sha-1 & Merkle-Damgard Construction

Sanjeev Kumar
School of Computer Science
Lovely Professional University
Phagwara, Punjab-144401

Abstract

Realness and privacy assumes an imperative job for correspondence. In order to provide security to the information over correspondence medium, we have numerous cryptography related calculations, hash capacities, and so forth. We are proposing a hash work by joining SHA-1 and MD5 which gives preferable security over the others. There has been different aspects of SHA algorithms and their workings in providing security and reliability. SHA-1 and Merkel Damgard are used to construct a more reliable and secure algorithm for providing cryptographic solutions.

Keywords: secure hash algorithm, Message Digest,Security

1. Introduction

As innovation is improving step by step in the fields of correspondence and data frameworks, a proficient use of electronic gadgets and media is used for communicating between two ends[1-2].In this process, the user will simply enter the text which is simply being divided into the equal blocks of length of 512 bits (naming x_1, x_2 ,x_3……..x_n). If the block length is less than 512 bits, the algorithm will pad with some bogus bits (padding will be shown later). In the next step, the data will be input inside the compression function where a compression technique will be applied on the blocks. At the end of this step, the very first hash value for x_1 with IV vector. This calculated hash is represented by z_1. The $z_1$value will again supplied back into the compression function, which becomes the new IV vector for x_2 and gets concatenated with the next value for x which will produce the next hash value. This process will continue for z_i values, where i=n. In this way the whole data blocks will be processed *i* iterations. Finally, we calculatez_i=H(x), where H(x) is the hash of "x" [3-4]. The whole process is shown below.

2. Review of Literature

a. Public key cryptography:

In the public key cryptography, there are two different keys which will be used for encryption and decryption separately [6]. These keys are known as private key and public key. The private key will be kept secret by the sender and the public key will be available in the public domain. The encryption process can be described as follows: the sender will take the receiver's public key from the public key directory as it is available to all or open in the public domain and will

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

use that key to encrypt the data and producesthe cipher text. This particular cipher text will be shared on the insecure channel and will be received by the receiver on the other side. The receiver will use his private key which is kept secret to decrypt the cipher-text to obtain back the plain-text [7-8]. So in this cipher technique, there are two separate keys which will be used to encrypt and decrypt thetext separately. Therefore, these types of cipher techniques are much more useful and hard to break down.

A cryptographic hash function or simply the hash function is a kind of function that actually takes a random block of data, and returns back the fixed size bit string. It is often represented by H(x), where x is a random size block message [9].

b. Symmetric key cryptography

The symmetric key algorithm uses the same key for encryption and decryption [10-12]. For example, Alice wants to send some data to Bob, but she wants data to be secured enough, that it should not get compromised. Now, whatever the algorithm is being created is in the public domain, but the main responsibility is of the key (which is kept secret) to make the data secure [13-16]. In this particular cryptographic algorithm, both the sender (Alice) and receiver (Bob) agreed upon a key, and using that key Alice will first encrypt the data whatever the data she wants to send. After encryption,the cipher text is being sent over an insecure channel. After receiving the cipher text by Bob, it will be decrypted using the same key and plain text is recovered.

3. Markel Damgard construction algorithm:

MD construction is based on the iterative hash function methodology just leaving the fact that it uses the compression function which produces a hash which is collision resistant.

Some points of Markel Damgard Construction

1. MD construction uses an iterative hash function methodology

2. The compression function used as:

Compress (C) : {0,1} pow (m+t) □ {0,1} pow (m), where compress (C) is a compression function, m is the actual required bits, t is the padded bits, {0,1} are the bit representation of the actual message x.

3. This is actually used to construct the hash functions which are also free from the collision resistant attacks.

H: {0,1} pow (*) --> {0,1} pow (m), * represents the iterations.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

This construction yields the proof of the above results because it gives the very wonderful results for whatever the query being asked.

Steps for construction in developing the SHA-1 or any other series of SHA family of hashes using this Markel Damgard Construction:

Let's consider the message "x" which is quite large, and using that it's hash value has to be constructed.What's happening is that, a user is imputing the text of the variable length means it could of any length. It's going inside the SHA-1 algorithm some operations happened and it's giving the output of 160-bits as Y.

But how this is happening this? What's inside this? How we are getting this output of very short range? What could be possibly inside this algorithm?

All answers are inside this SHA-1 algorithm. As important, if we carefully look and think that whatever the data is going inside this SHA-1 function is getting compressed, so there will surely a compression function working inside SHA-1.

What's happening is that, a user is inputing the text of the variable length means it could of any length. It's going inside the SHA-1 algorithm some operations happened and it's giving the output of 160-bits as Y.

But how this is happening this? What's inside this? How we are getting this output of very short range? What could be possibly inside this algorithm?

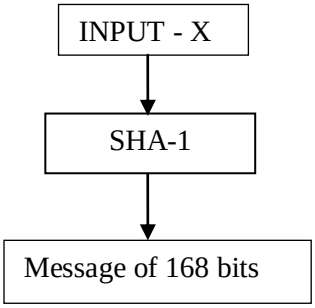

Figure 3(a): steps of SHA-1

All answers are inside this SHA-1 algorithm. As important, if we carefully look and think that whatever the data is going inside this SHA-1 function is getting compressed, so there will surely a compression function working inside SHA-1.

SHA-1 algorithm consists of 80 rounds to calculate hash value. The initial feed to the hash is a message (say X) by splitting it into various blocks. The following image depicts the steps.

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

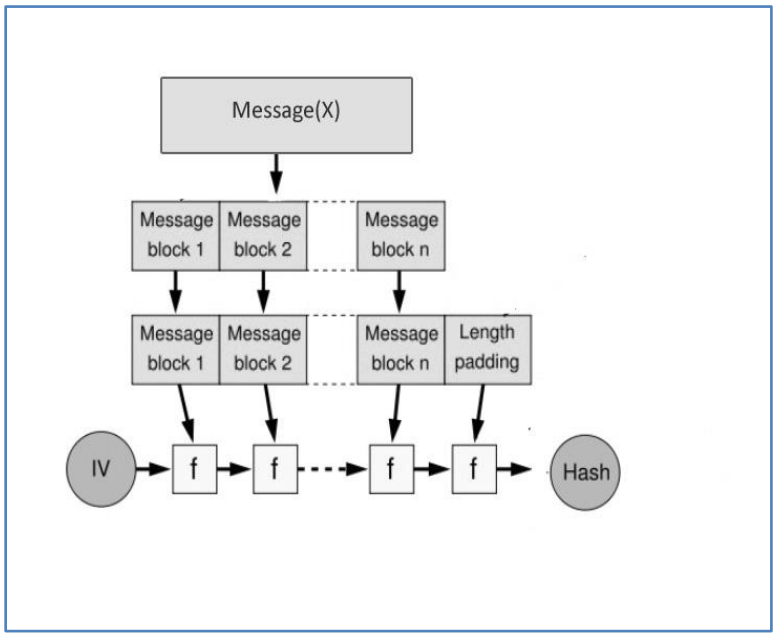

Figure 3(b): Calculation of Hash value

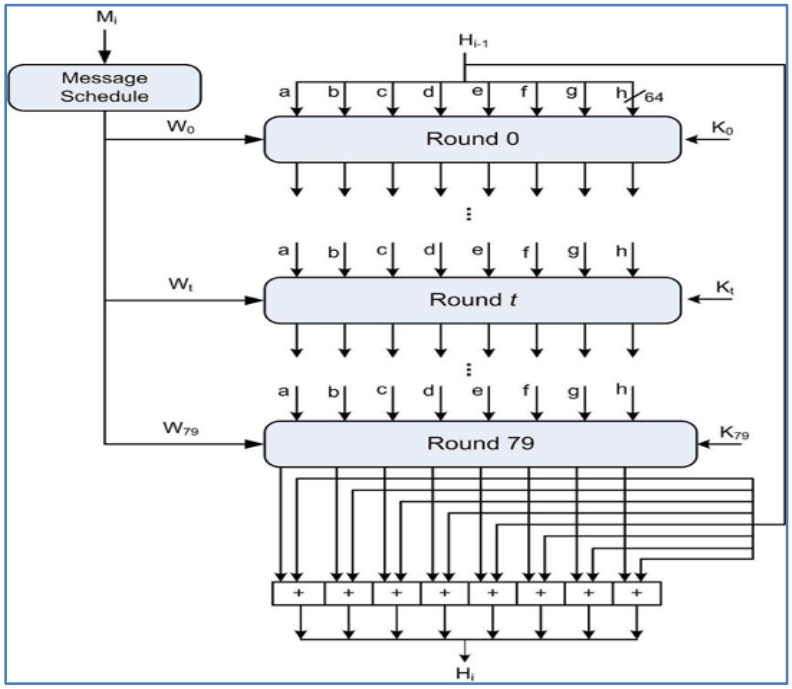

Figure 3(c): 80 Rounds in SHA-1

The SHA-1 takes a variable length messageand it delivers a length of fixed size message digest limited to 160 bits. In the above outlined figure, it is demonstrated that the message M, is isolated into singular squares and after the separating of those into singular piece hinders, the

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

information is added with the cushioning bits, passed it through apressure work with the two data sources which at last creates a hash value as (H).

4. Result

As appeared above, the complete 80 rounds will be occurred in SHA-1 for producing of hash value. The capacity of SHA-1 is indistinguishable from SHA-0 expect for the message development where message turn is required. It very well may be appeared as the equation given below

$$W_t = \begin{cases} m_t & \text{for } 0 \le t < 16, \\ RL(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}, 1) & \text{for } 16 \le t < 80. \end{cases}$$

In our rotation function, the inserted data will be left rotated for the expansion of data. The steps have been shown as following:

```
Initial vector: 0x0c860b4f 0x49efb38e 0xe5a4eb5c 0x87b01c18
Searching for sha collision (this can take several hours)...
Random seed: 1562188784
Progress: 38.434.1337.49
Collision blocks:
unsigned int m0[32] = {
0x02a4ae92, 0x6537a4d7, 0x13f5b5e1, 0xd1bca5e9,
0x4a8b4f42, 0xc42dfdcd, 0xf4149ecb, 0xa941d730,
0x05336cc1, 0x02f345ff, 0x8b76eb37, 0x53226922,
0x7fd9f4e4, 0x4cde5ff2, 0xd79d8e07, 0x7fa04148,
0x95ffe390, 0xea92d0a9, 0xcef0a708, 0xc2eaadd0,
0xc4ee17ca, 0x0cc63bca, 0x44e2393c, 0xfdedc94c,
0x9f32a805, 0xe6b25d93, 0x5fe31ebe, 0x273d80ae,
0x95442509, 0x317bb9d1, 0x69729b53, 0xf1fcd239,
};

unsigned int m1[32] = {
0x02a4ae92, 0x6537a4d7, 0x13f5b5e1, 0xd1bca5e9,
0xca8b4f42, 0xc42dfdcd, 0xf4149ecb, 0xa941d730,
0x05336cc1, 0x02f345ff, 0x8b76eb37, 0x5322e922,
0x7fd9f4e4, 0x4cde5ff2, 0x579d8e07, 0x7fa04148,
0x95ffe390, 0xea92d0a9, 0xcef0a708, 0xc2eaadd0,
0x44ee17ca, 0x0cc63bca, 0x44e2393c, 0xfdedc94c,
0x9f32a805, 0xe6b25d93, 0x5fe31ebe, 0x273d00ae,
0x95442509, 0x317bb9d1, 0xe9729b53, 0xf1fcd239,
};
```

Figure 1: Rotation Function

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

The result of the working SHA-1 and MD5 has been obtained by implementing it in Linux environment. By analyzing the result, we can indicate that the IV vector(instatement vector) as: [0x0c860b4e 0x49efb38f 0xe5a4eb5a 0x87b01c13].

The output here which we have obtained is in the form of 32-bit unsigned integer.

5. Conclusion

We can conclude here that SHA-1 takes on an arbitrary length of data and gives back the 128-bit hash value. Rest of the working is same as the SHA-1. It is based on the Markel Damgard construction and also based on the iterative manner to produce hash. For padding, it divides the file to 64 bytes of equal length of blocks and it appends the value to 448 modulo 512 and takes 4 words (A,B,C,D) to process the round functions.The results shows that the proposed solution will be better in providing security and reliability.

6. References

[1]. Erfaneh Noroozi, Salwani Mohd Daud, Ali Sabouhi: "Secure Digital Signature Schemes Based on Hash Functions", IJITEE, Volume-2

[2]. W. Diffie, M. Hellman, "New directions in cryptography" Information Theory, IEEE Transactions on 22.6, pp. 644-654, Jun 1976

[3]. Ilya Mironov; "Hash functions: Theory, attacks, and applications" Microsoft Research, Silicon Valley Campus. 2006

[4]. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD", EUROCRYPT, pp. 1–18, 2005.

[5]. John Kelsey and Bruce Schneier, "Second Preimages on n-Bit Hash Functions for Much Less than 2n Work", EUROCRYPT, Springer, pp. 474–490, 2005.

[6]. W. Stalling "Cryptography and Network Security Principles and Practices", Prentice Hall, Fourth Edition, 2005, P 353.

[7]. Biham, E., Shamir, "A.: Differential Cryptanalysis of the Data Encryption Standard". Springer, Heidelberg (1993)

[8]. Biham, E., Chen, R.: "Near collision for SHA-0" .In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)

[9]. Chabaud, F., Joux, "A.: Differential Collisions in SHA-0". In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)

ISSN: 0971-1260
Vol-22-Issue-3-July-2019

[10]. "National Institute of Standards and Technologies", Secure Hash Standard, Federal Information Processing Standards Publication, FIPS-180 (May 1993)

[11]. National Institute of Standards and Technologies, Secure Hash Standard, Federal Information Processing Standards, Publication FIPS-180-1 (April 1995)

[12]. Rivest, R.: The MD4 Message-Digest Algorithm, Network Working Group, Request for Comments:1186 (October 1990)

[13]. Rivest, R.: The MD5 Message-Digest Algorithm, Network Working Group, Request for Comments:1321 (April 1992)

[14]. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis for Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)

[15]. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

[16]. Rijmen, V., Oswald, E.: Update on SHA-1. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 58–71. Springer, Heidelberg (2005)