Vol. 4, Issue 2 (Apr. - Jun. 2016)
ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Secure Hash Design & Implementation Based On Md5 & Sha-1 Using Merkle – Damgard Construction

Shaik.Mohammed Sharief
School of Computer Science and Engineering, Lovely Professional University, Punjab, India

Abstract
From a long time ago, to till now, authenticity and confidentiality plays a vital role in the communication. For providing the security for the data over communication medium we have many cryptographic algorithms, hash functions, etc. Here we are designing a hash function by combining SHA-1 and MD5 which provides better security than the individual. We also reviewed the definitions, different types of attacks, different types of hash functions and the design principles& working of the proposed scheme.

Keyword
Secure Hash algorithm, Message Digest, Security, Hash Function, Cryptography.

I. Introduction

As technology is developing day-by-day in the fields of communication and information systems, we are making efficient utilization of electronic devices and media for data transfer and for communicating between two parties. In these information systems and communication channels large amount of data is being sent. There is a need to protect this data from the intruder who tries to get the data from the communication channel and tries to steal the secure communication between the parties. For securing the data many encryption algorithms have been invented and used successfully. But, the attacker who is aware of these cryptographic functions that are used to encrypt the data can easily decrypt the data and can steal the information. For example, the most widely used AES encryption algorithm has also been attacked bliclique attack and successfully recovered the key. For AES-128, the complexity for recovering has been given as $2^{126.1}$. So, there is a need for providing more security other than usage of cryptographic algorithms for enciphering the data.

Hash function was introduced in the field of cryptography for providing better security. This tool is used to provide authenticity for the data over the communication medium. This hash function will take an arbitrary block of data and returns a fixed length of bit string called the hash value. This hash value can also be called as message digest and the data that we are going to encrypt is called as message. The message digest is created by using an algorithm from a file which uniquely represents that file. If the attacker attacks the file and changes the data, the message digest will also change, which can be easily identified by the receiver when he calculated the message digest. The other applications of MD include providing e-mail security and CRC for files, etc.,

In this paper, we are going to explain about the SHA and MD algorithms, the previous works in the next section. It is followed by the Implementation of SHA-1 with MD5. The following sections discuss the implementation and results analysis followed by the conclusion.

II. Review of Literature

The cryptographic hash functions have many applications in the field of information security such as Message authentication codes, Digital Signatures and other forms of authentication. These algorithms take an arbitrary length as an input message and provide the fixed length output.

Input:

$$M = \left((A-Z)(a-z)(0-9)(?\,,!)\right)^{*}$$

The input equation defined above tells that the input string may contain and number of characters from defined set of values. The '*' defines that the length of the string is variable.

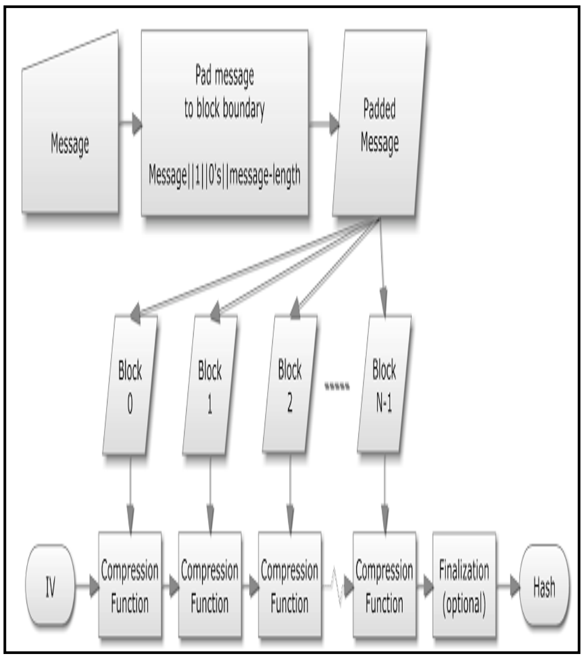
Output:

$$M = \left((A-Z)(a-z)(0-9)(?\,,!)\right)^{n}$$

The output equation defines that the input given which is of variable length is converted into the fixed length data which is indicated by using "n" in the equation. For providing the fixed length message digest we are using SHA-1 algorithm. It is most widely used of the existing SHA hash functions and employed in several applications and protocols. It provides the message digest based on the principles developed by Ronald L. Rivest in the design of MD5. It is most widely used in protocols including SSL, PGP, S/MIME etc. The construction of SHA-1 algorithm is same as MD5 but varies in producing the message digest where, MD5 produce 128 bit message digest and SHA-1 produce 160-bit message digest.The algorithm takes the input as 512 bit blocks and has 80 numbers of rounds. Here, we are using Merkle – Damgard construction which is most widely used hash function on which most of the hash functions and standardized hash functions are used.

III. Algorithm

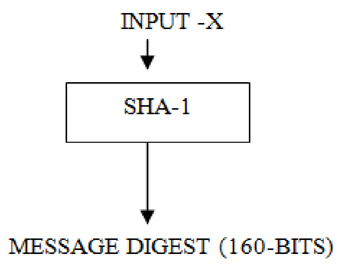
Here, we are using the Merkle – Damgard function for construction of the hash function.The Merkle – Damgard function is based on the iterative hash function and it uses a compression function which prevents the hash from the collision. The construction using the Merkle – Damgard function can be illustrated as:



In this MD construction we are using the compression function which reduces the size of output when compared with the output. The compression function is repeatedly used to process the entire message. The construction builds a hash function based on the compression function which takes two inputs in which the first one is changing Intermediate Hash Value of K bits and the second one is Message block of N bits. An input message is padded with single '1' bit followed by the number of '0' bits requires to obtain original length of the bit blocks. The padded message is now split into blocks of size exactly N-bits. Now the hash function will starts from initial hash value (IHV) and for subsequent blocks it calls the compression function with current and Message block and stores the output as . After processing of all the blocks, the final output will be produced.

3.1 Construction of SHA using MD

For SHA-1 we are now considering a message of length 'X' which is large in size. The SHA-1 will take the message of variable length and provides output of 160 bits hash value.

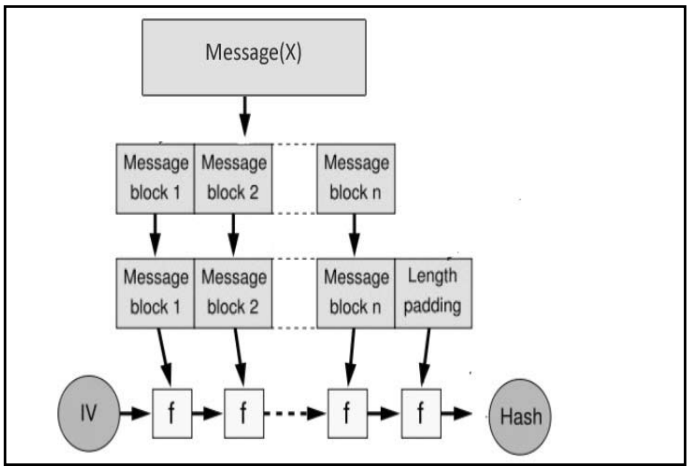


The SHA-1 takes a variable length message and produces a fixed length output of message digest finite to 160 bits is shown as above.

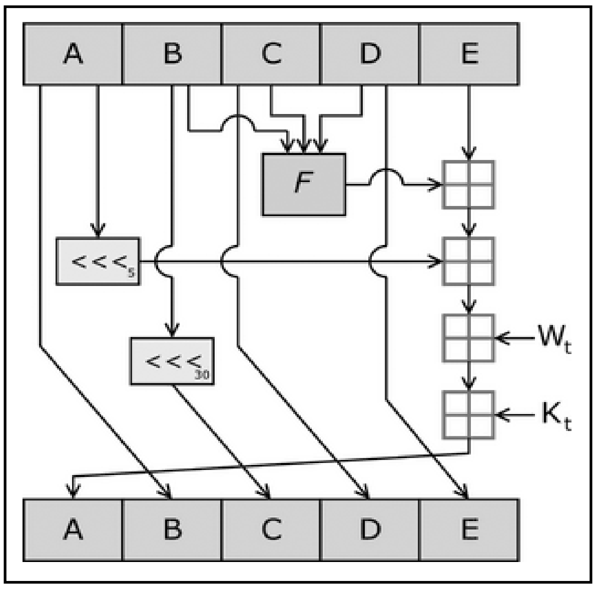
In this after the user enters the text, it will be divided into 512 bit blocks of equal length and the last block if it is less than 512 bits it will be padded with one '1' bit followed by number of '0' bits to achieve 512 bit block. Each of these 512 bit blocks are named as (X_1, X_2, X_3, … X_{i+1},…,X_n)

Each of these bit blocks (X_i) will go under compression function along with an Initial Vector (IV) and produces an output of compressed data (C_i). Now, this C_i is given as a feedback to the compression function again to act as a input to the next vector X_2. This process will continue until the all the blocks of data till X_n are converted into C_n. The final outcome C_n will be treated as:

$$C_n = H(x)$$



Where *H* indicates hash value for the block (x).

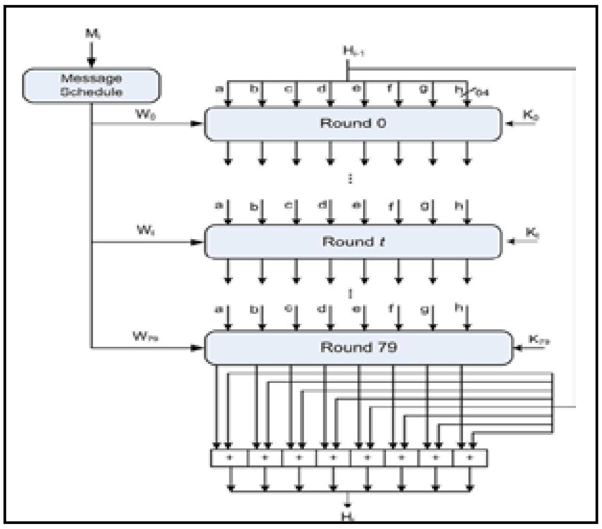


In the above illustrated figure, it is shown that the message M, is divided into individual blocks and after the dividing of those into individual bit blocks, the data is appended with the padding bits and later Passed it through a compression function with the two inputs which finally produces a Hash Value (H)

As shown in above, the total 80 rounds will be taken place in SHA-1 for generating of Hash Function. The compression function of SHA-1 is identical to SHA-0 expect for the message expansion where message rotation is needed. It can be shown as

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

In the rotation function the data will be left – cyclic rotated for the expansion of data. It can be shown as:

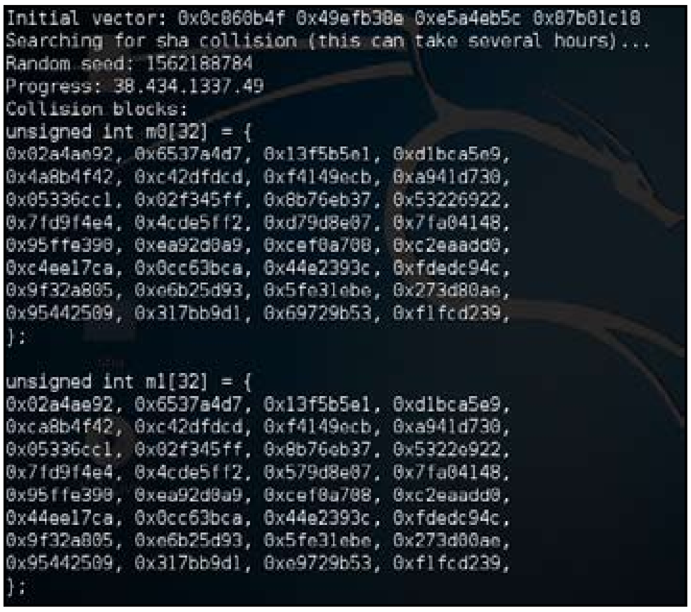


The above illustration shows the single round details of the SHA-1 constructed by Merkel – Damgard. This process will be taken in 80 rounds of SHA-1 for the formation of Message Digest.

IV. Result

The result has been obtained by programming in it Linux environment. In the result we have shown that initialization vector (IV) as:
0x0c860b4f 0x49efb38e 0xe5a4eb5c 0x87b01c18
The output we have obtained in the form of 32-bit unsigned integer. It can be seen as:
As shown in above, the total 80 rounds will be taken place in SHA-1 for generating of Hash Function. The compression function of SHA-1 is identical to SHA-0 expect for the message expansion where message rotation is needed. It can be shown as

In the rotation function the data will be left – cyclic rotated for the expansion of data. It can be shown as:



```
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,
0x95ffe398, 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,
0x95ffe398, 0xea92d0a9, 0xcef0a708, 0xc2eaadd0,
0x44ee17ca, 0x0cc63bca, 0x44e2393c, 0xfdedc94c,
0x9f32a805, 0xe6b25d93, 0x5fe31ebe, 0x273d80ae,
0x95442509, 0x317bb9d1, 0xe9729b53, 0xf1fcd239,
};
```

V. Conclusion

This paper contains the details about the implementation of SHA-1 using the Merkel – Damgard Function which uses compression technique to avoid hash- collision. In this we have taken SHA-1 algorithm for Hash value generation. In future this can be extended for other algorithms of SHA series. This algorithm provides efficient security and also the solution for collision of Hash.

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.*