ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Modifying SHA-512 using Padding, Tree structure and Permutation Boxes

Sumita Tyagi
[*] Computer Science & Engg., Babu Banarsi Das Institute of Technology, India

Abstract

In this age of technology internet has become a day to day necessity. With this growing usage of web data it has become very important to design some secure way to save this web data from various types of attacks. One of the ways to resolve this problem is hashing. Hashing takes variable length input and converts it into a fixed length output using various hash algorithms like MD4, MD5 and SHA. Despite of all these advances in the field of hashing these algorithms still have some weaknesses and many attacks on these algorithms have been discovered. In this paper we propose a new way of hashing applied to the traditional algorithms. The basic concept is same but we have modified the few steps involved in the process. We combine padding using SALT, tree structure and permutation boxes applied together in these algorithms. This helps to provide an output that is more secure and complicated making it more resistant to various collision attacks. Possibility of collision attacks, rainbow table attacks and birthday attacks is also mitigated by the complex structure of the algorithm.

Keywords: SALT, Permutation Boxes, Rainbow Table Attack, Tree Structure

Introduction

The secure hash function algorithm (SHA) was developed by the national institute of standard and technology (NIST) in 1993. SHA-1 produces a hash value of 160 bits [5]. In 2002 , NIST produced a revised version of the standard , FIPS 180-2 , that defined three new versions of SHA , with hash value length of 256 , 384 , 512 bits known as SHA-256 , SHA-384 , SHA-512 . These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operation as SHA-1. Actually a cryptography hash function is a procedure that takes an arbitrary block of data and returns a fixed–size bit string known as hash value, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message", and the hash value is sometimes called the message digest or simply digests**.** In 2005, NIST announced the intention to phase out approval of SHA-1 as it was prone to many attacks [7] and move to a reliable SHA version of 2010. There and then SHA-512 came into picture and is being widely used. Still SHA-512 also had some drawbacks and in meantime various attacks had been discovered on SHA-512. Many modifications to SHA 512 have also been made to make it more secure [3] [8].

Properties of hash function

1 These hash functions can be applied to any size data producing a fixed-length output.
2 Hash functions H(x) are relatively easy to compute for any given message *x*
3 Follows one-way property where it is computationally infeasible to find *x* such that H(*x*) = *h*
4 Have weak collision resistance where it is computationally infeasible to find $y \neq x$ such that H(*y*) = H(*x*)
5 Hash functions are strong collision resistance computationally infeasible to find any pair (*x*, *y*) such that H(*x*) = H(*y)*

Algorithms at a glance

SHA-1

SHA was designed by NIST & NSA in 1993, revised in 1995. It produces 160-bit hash value. SHA 1 was based on the design of MD4 with key differences. It pads the message so that its length is congruent to 448 mod 512 by appending a 64-bit length value to

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

the message. Then initialize 5-word (160-bit) buffer (A,B,C,D,E) to (67452301,efcdab89,98badcfe,10325476,c3d2e1f0) . After this process the message in 16-word (512-bit) chunks: expand 16 words into 80 words by mixing & shifting, use 4 rounds of 20 bit operations on message block, buffer and finally add output to the input in order to form new buffer value. The output hash value is the final buffer value

SHA 512

1. **Append the bits with padding:** 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 [length mod 1024 = 896). The padding is usually 1 followed by many 0's.

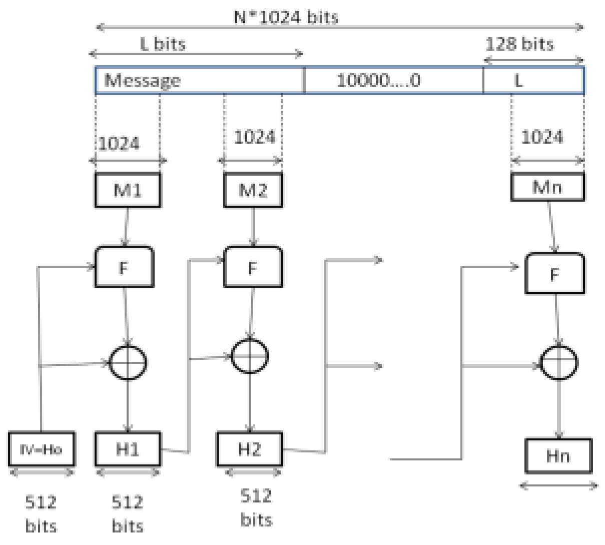

Figure 1. General Structure of SHA 512

2. **Append length.** A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding)
3. **Initialize buffer.** A 512-bit buffer is used to hold intermediate and final results of the hash function.
4. **Process message in 512-bit (16-word) blocks.** The heart of the algorithm is a module that consists of 80 rounds of processing. The 80 rounds have the same structure, but vary some constants and logical functions.
5. **Output.** After all *N* 1024-bit blocks have been processed; the output from the *nth* stage is the 512-bit message digest.

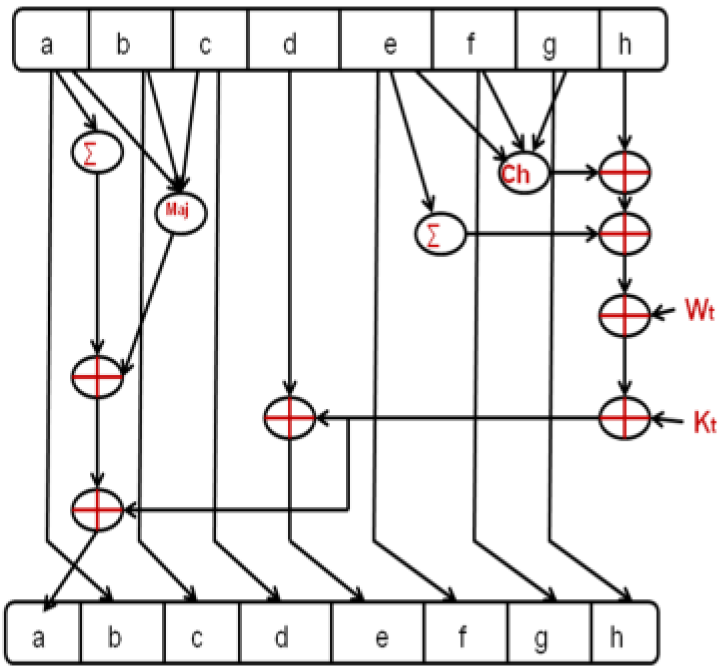

Figure 2. Single Round function of SHA 512

In this algorithm the result of the previous stage is added on to the next stage till the nth block of message is reached. As the new result depends on the previous result it increases the complexity of this algorithm.

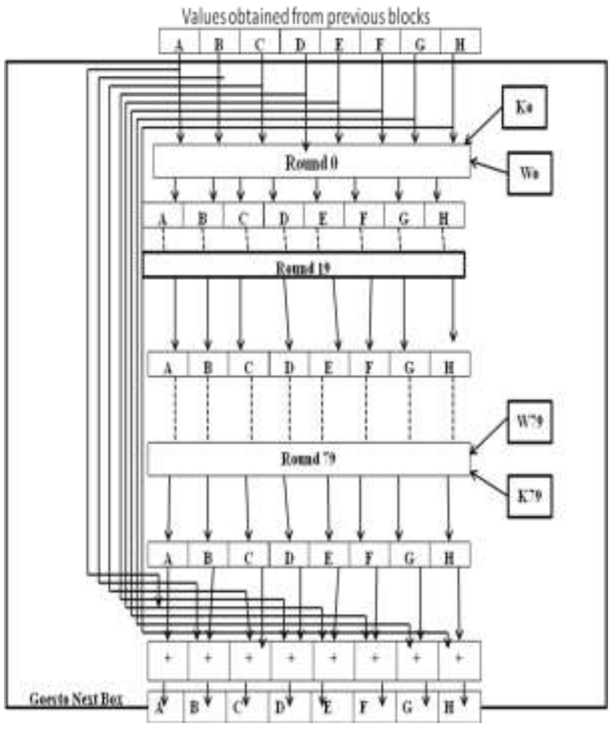

Figure 3. SHA 512 Compression Function

Proposed algorithm

SHA 512 is also prone to much different type of attacks [2] amongst which common ones are Birthday attack [4] and Rainbow Table Attack. We have modified the SHA 512 structure by using the concept of iterative hashing [1], extra random padding and permutation boxes. We implement the iterative hashing by using a tree like structure as shown in Figure 4. It differs from original structure of cascade at single level by using tree structure. In this nodes at first level are same as in previous algorithm but at subsequent levels these are half than that at previous level. So a tree structure is formed that decreases down the levels. This tree structure ensures security at multiple levels, thus increasing the complexity and decreasing the chances of collision.

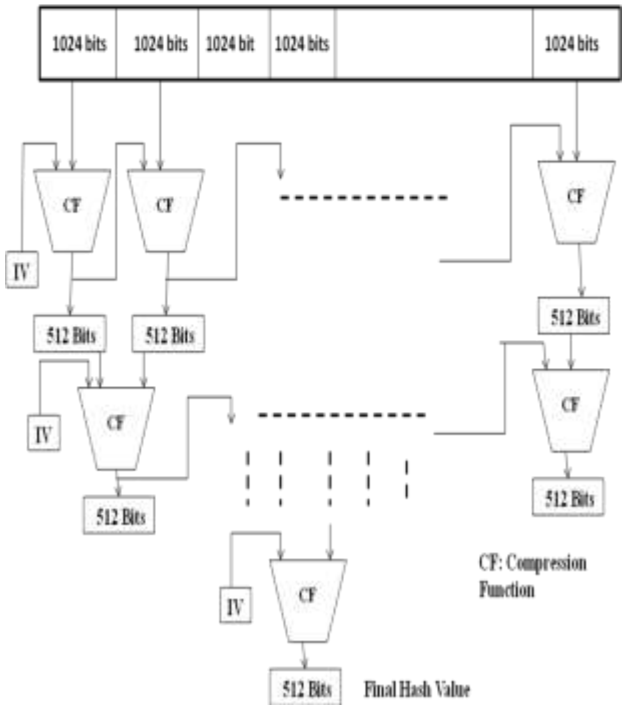

Figure 4. Tree Structure of SHA 512

The modified algorithm consists of the following steps:

1. Append padding bits:

While modifying SHA 512 we pad original message with salt which is variable and randomly decided by sender.

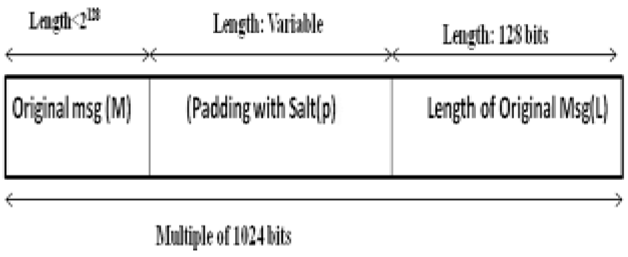

Figure 5. Adding Salt to Original SHA 512

Number of salt bits depends upon original message so as to make its total length, a multiple of 1024 bits. SALT is just an additional string appended to the password before it is being hashed and stored into the database table. The same exact SALT will be required during verification between the user entered password and stored password. SALTS are resistant to rainbow table and brute force attacks.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

2. **Append length:** A block of 128 bits is appended to the message. This block is treated as an unsigned 128 bit integer. The outcome of first two steps yields a message that is an integer multiple of 1024 bits in length.
3. **Initialize the buffer:** A 512- bit buffer is used to hold intermediate and final result of the hash function. the buffer can be represented as 64- bit registers(a,b,c,d,e,f,g,h)

a=6A09E667F3BCC908
e=510E527FADE682D1
b=BB67AE8584CAA73B
f=9B05688C2B3E6C1F
c=3C6EF372FE94F82B
g=1F83D9ABFB41BD6B
d=A54FF53A5F1D36F1
h=5BE0CD19137E2179

These words were obtained by taking the first 64 bit of fractional parts of square roots of first 8 prime numbers.
4. **Process message in 1024bit blocks:** The algorithm consists of 80 rounds, after every 20 rounds permutation of registers take place. Permutation after every 20 rounds depicts the use of switch functions which is a concept adapted from DES and thus will increase the security of SHA 512. Each round makes use of a 64 bit value W_t, derived from current 1024 bit block being processed(M_i), each round also makes use of an additive constant K_t. These words represent the first 64 bits of the fractional parts of the cube root of the first eighty prime numbers. Output of 80th round is added to the input to first round (H_{i-1}) to produce H_i, addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1}, using addition modulo 2^{64}.

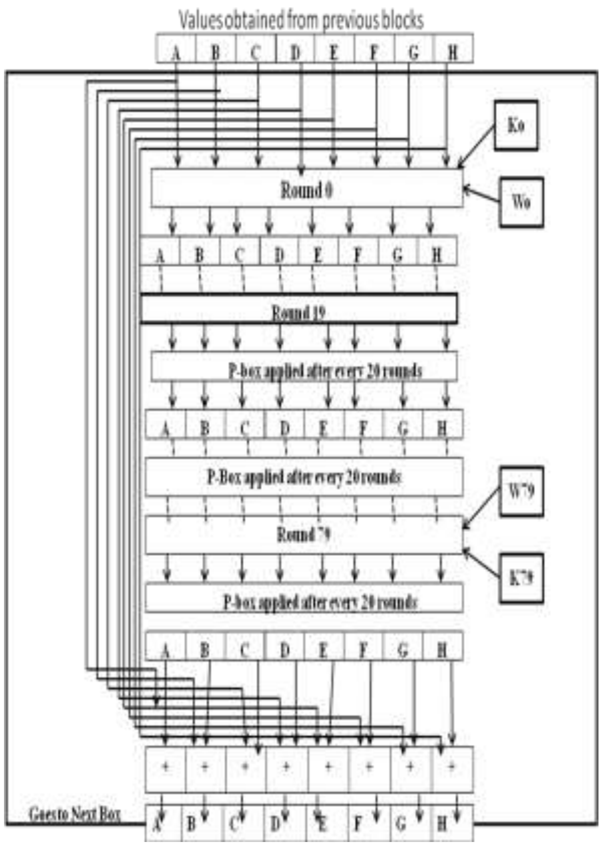
Values obtained from previous blocks
A
B
C
D
E
F
G
H
K0
W0
Round 0
Round 19
P-box applied after every 20 rounds
P-Box applied after every 20 rounds
W79
Round 79
K79
P-box applied after every 20 rounds
Goes to Next Box

Figure 6. Modified Compression Function of SHA 512

5. **Output:** After all N 1024 bit blocks have been processed the output from the stage is combined together so as to act as input to the next stage which proceed as the previous one. These results in single 512 bit message digest at the end of the tree.

Results and discussion

The algorithm proposed in this paper makes use of the salt function which is added to the passwords before these are hashed. This makes the hash function generated to be resistant to any kind of rainbow table attack or brute force attack or the collision attacks [7]. After this we have permuted the output obtained after every 20 round using permutation boxes same as in DES. By using P Boxes the output obtained from the various blocks of input becomes more complicated and changing a single bit of input changes many bits in output hash function thus making it again collision resistant. Finally a tree structure is formed by combining various 512 block output and iteratively adding these to next block of

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

input. This result in forming only a single 512 bit output as a hash function. As many times iterative hashing is done this provide more security to SHA 512 algorithm.

Conclusion

There are many new and recent designs developed in hash algorithms [6] [8]. Amongst all the hash algorithms developed SHA 512 is a more secure algorithm. We have introduced three concepts in the design of SHA 512 by adding SALT, Tree structure and Permutation Boxes. These three changes to already existing SHA 512 makes it more complex and secure and prevents it from many attacks theoretical as well as practical that have been discovered on SHA 512.

References

1. E. Biham O. Dunkelman " A framework for Iterative Hash functions" Proceeding second NIST workshop2006, Santa Barbara, USA, August 2006.
2. Marc Martinus Jacobus Stevens "attacks on hash functions and application"Centrum Wiskunde & Informatica March 2010.
3. Mohammad Abu Taha, Mousa Farajallah, Radwan Tahboub: A Practical One Way Hash Algorithm based on Matrix Multiplication, International Journal of Computer Applications (0975 – 8887)Volume 23– No.2, June 2011
4. Erika Batista, Gaël Canal, Karim Ziadeh: The Birthday paradox Operational Research and Optimization, December 2012.
5. Dai Zubin,Zhou Ning:FPGA Implementation of SHA-1 Algorithm,IEEE-2003,pp 971-975.
6. Rajeev Sobti, G. Geetha "Cryptographic Hash Functions: A Review" IJCSI, Volume 9, issue 2, March 2012, ISSN (online) : 1694-0814
7. X. Wang, Y.L. Yin, H. Yu, "Finding collisions in the full SHA-1," Advances in Cryptology, Proceedings Crypto'05, LNCS 3621, V. Shoup, Ed., Springer-Verlag, 2005, pp. 1–16.
8. S.Al Kuwari, J.H. Davenport, R.J. Bradford. Cryptographic Hash Functions: Recent Design Trends and Security Notations. In short paper proceedings of Inscrypt'10. Science Press of China, 2010, pp- 133-150.

Author Biblography

	Sumita Tyagi Sumita Tyagi has received her B.Tech Degree in Information Technology from Ideal Institute of Technology in the year 2006 and her MBA in HR and IT from IMT in the year 2009. Currently working as Asst. Prof. in BBDIT, Ghaziabad and has more than 7 yrs of experience in field of academics. She has authored a book on "Cryptography & Network Security". Areas of interest include Cryptography, Algorithms and Programming. Email: sumita.tyagi12@gmail.com