IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. VII (Sep – Oct. 2014), PP 125-129
www.iosrjournals.org

Enhanced Message Digest Version 5 Architecture for Secure Hashing

Mwangi Joseph, Teresah Wachiuri, Dr. Wilson Cheruiyot

Abstract: *The message digest algorithm is a widely used cryptographic hash function. It produces a 128-bit hash value. It has been used in a variety of security applications and is also commonly used to check data integrity. However MD5 has been found not to be collision resistant. It is possible to find a pseudo-collision, that is, two different initialization vectors which produce an identical digest. Other attacks that have been found to be working on MD5 are rainbow, dictionary and brute-forcing attacks. The ability to find collisions has been greatly aided by the graphics processing unit (GPU). The solution to these collisions is a robust MD5 compression function. This research paper focused on enhancing the Merkle Damgard model construction, which is the core component of the hashing process. It involved adding an extra iteration so that instead of the current two -cycle looping, we have a three-cycle looping per message block. The aim is to make it computationally infeasible to reverse the hash function. In this way, make the whole MD5 algorithm would be made stronger. This algorithm could then be used for password hashing in an effort to make them robust. The new password hashing mechanism find applications in areas such as digital signatures and digital certificates.*
Keywords: *Hashing, MD5, GPU, Architecture, MITM*

I. Introduction

Many software services provide an authentication system that relies on a user name and password combination. Initially, this password is generated by the user and stored in a safe location on the system (Ness, 2012). To make sure that these passwords are still safe even if the security of the location cannot be guaranteed, it is common to use a cryptographic hash function to calculate the digest of the password and store this together with the users' credentials. When a user authenticates himself to the system again, the digest of the plaintext password is calculated and compared to the stored digest (Sprengers, 2011).

MD5 hashing scheme takes as input a message of arbitrary length and produces as output a 128 bit fingerprint or message digest of the input (Harley, 2003). It was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4. An MD5 hash value is typically expressed as a hexadecimal number, 32 digits long. Its compression function is based on Merkle Damgard construction. In 1989, Damgard and Merkle independently proposed a similar iterative structure to construct a collision resistant cryptographic hash function:

$$H : \{0, 1\}* \longrightarrow$$

$\{0, 1\}^t$ using a fixed length input collision resistant compression function

$$f : \{0, 1\}^b \times \{0, 1\}^t \rightarrow \{0, 1\}^t.$$

Since then, this iterated design has been called Merkle-Damgard (MD) construction which influenced the designs of popular dedicated hash functions such as MD5, SHA-0 and SHA-1. The design motivation of the MD construction is that if the compression function *f* is collision resistant then so is the resultant iterated hash function *H*. It is known that, a compression function *f* secure against the fixed initialization vector (IV) collisions is necessary but not sufficient to generate a secure hash function *H* (Jonathan and Lindell, 2008) .The latest multi-block collision attacks (MBCA) on the hash functions MD5, SHA-0 and SHA-1 prove this insufficiency(Mark, 2013). These attacks clearly show that these iterated hash functions do not properly preserve the collision resistance property of their respective compression functions with the fixed IV. There is therefore need to design a new architecture to improve the hashing process of MD5.

II. Related work

A study by Sebastiano (2012), gives a description of the Meet-in-the-Middle Framework (MITM) that can be used to descript an MD5 hash. Assuming $K = K_1 * K_2$ and $E_k = E^1k_1 \circ E^2k_2$, where E is the composition of two ciphers

whose keys are independent of each other. Considering a couple (m;Ek(m)), the computation of the cipher-text can be split as follows:

$$\begin{tikzcd} m \arrow[r, "k_1", "E1"'] & C' \arrow[r, "k_2", "E2"'] & c \end{tikzcd}$$

This implies that an attacker, given *m* and *c*, can perform the following computations:

$$\left\{\begin{array}{l} m \xrightarrow[E^1]{k_1} v_1 \quad \text{Ʉ}\, k_1 \text{€}\, K_1 \\ c \xrightarrow[D2]{k_2} v_2 \quad \text{Ʉ}\, k_2 \text{€}\, K_2 \end{array}\right.$$

If a match between any of the v_1 and any of the v_2 is found, the corresponding key (k_1; k_2) encrypts *m* to *c* . It is also possible to carry out a Splice-and-Cut , which is a more general form of MITM . This attack was used to prove that triple encryption under two different keys does not double the security level.

III. New Architectural Design

The MD5 can be split up into four parts: The first part is padding. At this stage, the message is padded with: the '1'-bit, next as many '0' bits until the resulting bit-length equals 448 mod 512, and finally the bit-length of the original message as a 64 -bit little-endian integer. The total bit-length of the padded message is 512N for a positive integer N. The second part involves partitioning. In this phase, the padded message is partitioned into N consecutive 512-bit blocks, M1,M2, . . . ,MN. The third phase is processing. In this stage, MD5 goes through N + 1 states IHV_i, for $0 \leq i \leq N$, called the intermediate hash values. Each intermediate hash value IHV_i consists of four 32-bit words a_i, b_i, c_i, d_i. For i = 0 these are initialized to fixed public values:

$$IHV0 = (a_0, b_0, c_0, d_0) = (67452301_{16}, EFCDAB89_{16}, 98BADCFE_{16}, 10325476_{16}),$$

and for i = 1, 2, . . .N intermediate hash value IHV_i is computed using the MD5 compression function described in detail as follows:

$$IHV_i = MD5Compress\ (IHV_{i-1}, Mi).$$

The last part is the output. The resulting hash value is the last intermediate hash value IHV_N, expressed as the concatenation of the sequence of bytes, each usually shown in 2 digit hexadecimal representation, given by the four $_{\text{string}}$words a_N, b_N, c_N, d_N using Little-Endian. For example, in this manner IHV_0 will be expressed as the hexadecimal

0123456789ABCDEFFEDCBA9876543210

The input for the compression function MD5Compress (*IHV,B*) is an intermediate hash value IHV = (a, b, c, d) and a 512 -bit message block B. There are 64 steps (numbered 0 up to 63), split into four consecutive rounds of 16 steps each. Each step uses a modular addition, a left rotation, and a non-linear function. Depending on the step t, an Addition Constant AC_t and a Rotation Constant RC_t are defined as follows:

$AC_t = |2^{32}\ |sin(t + 1)||,\ 0 \leq t < 64,$

$$(RC_t, RC_{t+1}, RC_{t+2}, RC_{t+3}) = \begin{cases} (7, 12, 17, 22) & \text{for } t = 0, 4, 8, 12, \\ (5, 9, 14, 20) & \text{for } t = 16, 20, 24, 28, \\ (4, 11, 16, 23) & \text{for } t = 32, 36, 40, 44, \\ (6, 10, 15, 21) & \text{for } t = 48, 52, 56, 60. \end{cases}$$

The non-linear function ft depends on the round:

$$f_t(X, Y, Z) = F(X, Y, Z) = (X \wedge Y)\ \overline{\ }\ (X' \wedge Z) \qquad \text{for } 0 \le t < 16,$$

$$G(X, Y, Z) = (Z \wedge X)\ \bar{\circ}\ (Z' \wedge Y) \qquad \text{for } 16 \le t < 32,$$

$$H(X, Y, Z) = X\ \bar{\circ}\ Y\ \bar{\circ}\ Z \qquad \text{for } 32 \le t < 48,$$

$$I(X, Y, Z) = Y\ \bar{\circ}\ (X \vee Z') \qquad \text{for } 48 \le t < 64.$$

The message block B is partitioned into sixteen consecutive 32 -bit words m0, m1, . . . ,m15 (using Little Endian byte ordering), and expanded to 64 words. The above functions, using the state variables and the message as input, are used to transform the state variables from their initial state into what will become the message digest. For each 512 bits of the message, the four rounds performed. After this step, the message digest is stored in the state variables (A, B, C, and D). To get it into the hexadecimal form, output the hex values of each the state variables, least significant byte first. For example, if after the digest:

A = 0x01234567;
B = 0x89ABCDEF;

C = 0x1337D00D
D = 0xA5510101
Then the message digest would be:

67452301EFCDAB890DD03713010151A5 (required hash value of the input value).

The current MD5 hashing scheme consist of 4 steps as depicted in Figure 1 below. The input to the MD5 is the initialization vector (IV), message blocks (labeled X1, X2…..XL) and the length bits L.

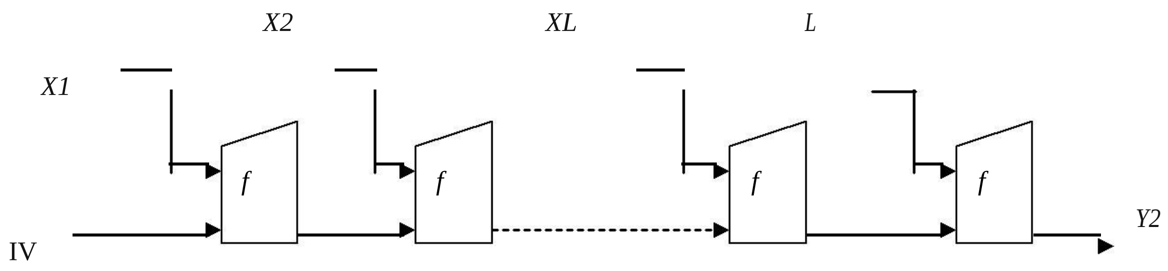

Figure 1: The Current MD5 Structure Y1

The output of the message block processing is labeled Y1, while that of length bit processing is L. These two outputs are then combined to produce the final hash labeled Y2.

IV. Result and Discussion

A collision resistant cryptographic hash function *H* following MD structure is a function that hashes a message *M* ∈ *{*0, 1*}*∗ to outputs of fixed length *{*0, 1*}t* . The specification of *H* includes the description of the compression function *f*, initialization vector (IV) state value and a padding procedure. Every hash function fixes the IV (fixed IV) with an upper bound on the size *|M|* of the input *M*. The message *M* is split into blocks *M*1*, . . .,ML*−1 of equal length *b* where a block *ML* containing the length *|M|* (MD strengthening) is added. Each block *Mi* is iterated using a fixed length input compression function *f* computing *Hi* = *f*(*Hi*−1*,Mi*) where *i* = 1 to *L* and finally outputting *HIV* (*M*) = *HL* as shown in Figure 2.

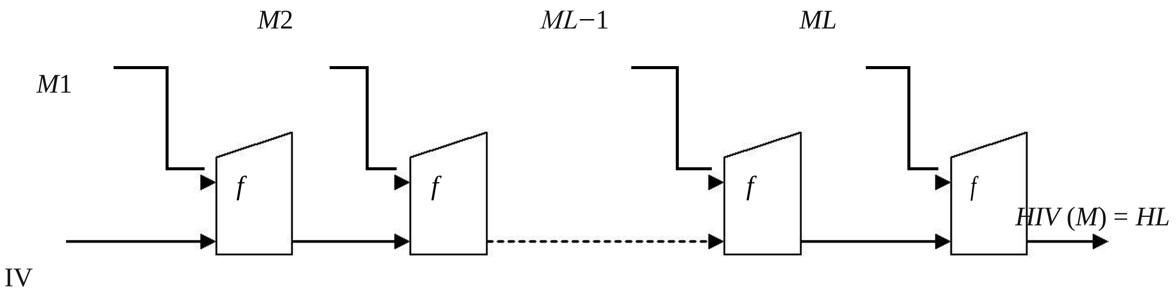

Figure 2: The Merkle-Damgard (MD) Construction

According to Sotirov (2012), a hash function *H* is said to be collision resistant if it is hard to find any two distinct inputs *M* and *N* such that *H*(*M*) = *H*(*N*). A hash function ⊕ *H* is said to be near-collision resistant if it is hard to find any two distinct inputs *M* and *N* such that *H*(*M*) *H*(*N*) = has some small weight. Based on the IV used in finding collisions, collision attacks on the compression functions are classified as follows (Stevens, 2012):

1. Collision attack: collisions using a fixed IV for two distinct messages. We call them *Type 1* collisions.

2. Semi-free-start collision attack: collisions using the same random (or arbitrary) IV for two distinct message inputs. We call them *Type 2* collisions.
3. Pseudo-collision attack: free-start collision attack using two different IVs for two distinct message inputs. We call them *Type 3* collisions.
As an example, suppose we are given the digest *H* of the message *M*, it is straight forward to compute *N* and *H'* such that *H'*= *H*(*M*||*N*) even for unknown *M* but for known |*M*|. The attack uses *H* (*M*) as the internal hash value to compute *H* (*M*||*N*).

This research paper adopted the Random Oracle Model (ROM). In the random oracle model, one assumes that some hash function is replaced by a publicly accessible random function (the random oracle). This means that the adversary cannot compute the result of the hash function by himself: he must query the random oracle. The random oracle model has been used to prove the security of numerous cryptosystems, and it has lead to simple and efficient designs that are widely used in practice (Patarin, 2012). The ROM is constructed as shown in Figure 3 below.

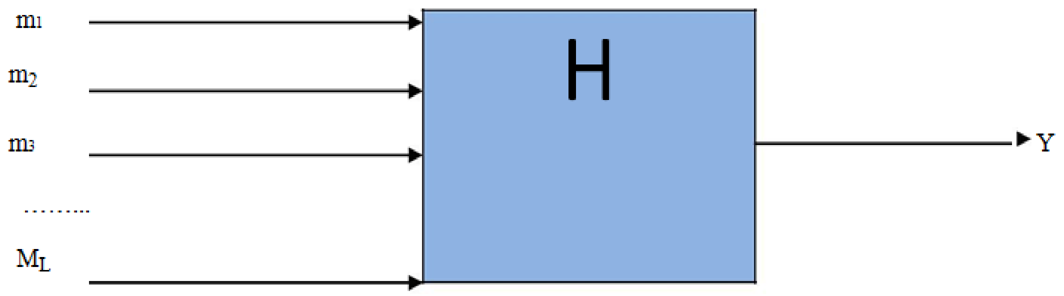

Figure 3: The Random oracle model (ROM)

The ROM was preferred due to its easy mapping to corresponding algorithms in programming languages where different modules can be handled by given classes and methods. Moreover, in ROM, security is often easier to prove, even for very simple and efficient schemes. This is because the simulator/security reduction has some extra flexibility: it gets to provide the view of the oracle to the adversary. This not only allows the simulator to know what queries the adversary makes, but also to program the oracle outputs that the adversary sees. Figure 4 below shows the working principles of the proposed architecture.

As can be seen in Figureb4, we included an *Vf chain* to the compression function. The total value of this chain was denoted by T. This value had to be padded with the length of the message to make it random and of fixed bits. The T value together with padded bits was denoted by T_p. An additional compression function *Vf* was then needed to handle the output of the *Vf chain* and the non-linear functions, f.

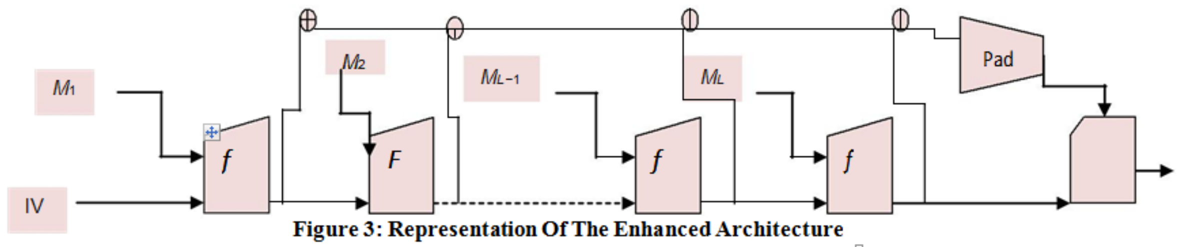

Figure 3: Representation Of The Enhanced Architecture

To process one block data, the compression function is executed three times; first to process the data block, next to process the padded block and finally the block *T* formed in the *Vf chain* as shown in Figure 3 above. While at least two blocks must be processed to find a multi-block collision on MD5, at least three blocks must be processed to create a multi-block collision in our new model. This increases the computational feasibility of finding collisions.

V. Summary And Conclusion

MD5 has is one of the most utilized cryptographic hash functions. It finds applications in several security applications and protocols. This includes verification of file transfers over the internet. However, it has been found to not to be collision resistant. While the collision resistance characteristic of the MD5 hash function has been broken, password hashing schemes based on this function, according to the study findings, can still be used securely. This emanates from the new architecture that was developed that could increase the time an anniversary could take to decrypt an MD5 hash. However, the researchers propose the use of new hash functions, such as SHA-3, that are more robust against birthday, Exhaustive search, Time-memory trade-off, rainbow, dictionary, reverse engineering Wang's attack and other similar attacks.

References

[1]. M. Sprengers, (2011), "GPU-Based Password Cracking", Masters thesis.
[2]. M.Stevens (2012), "Attacks On Hash Functions And Applications", PhD thesis, Universiteit Leiden.
[3]. K. Harley , (2003),"MD5 Algorithm".
[4]. D. Sebastiano,(2012), "Cryptanalysis Of Hash Functions", Masters Thesis, Technical University of Denmark.
[5]. S. Marc (2013), "Counter-Cryptanalysis", In Ran Canetti and Juan A. Garay, editors, CRYPTO (1), volume 8042 of Lecture Notes in Computer Science, pages 129{146.
[6]. J. Patarin,(2012), "The Random Oracle Model And The Ideal Cipher Model Are Equivalent", University of Luxembourg.
[7]. K. Jonathan and Y. Lindell (2008), "Introduction To Modern Cryptography", Chapman & Hall/CRC.
[8]. J. Ness, (2012), "Microsoft Certification Authority Signing Certificates Added To The Untrusted Certificate Store", TechNet Blogs, Security Research & Defense.
[9]. X. Wang and H. Yu ,(2005), "How To Break MD5 And Other Hash Functions", In Ronald Cramer, editor, Advances in Cryptology - EUROCRYPT, volume 3494 of Lecture Notes in Computer Science, pages 19–35.
[10]. A. Sotirov, (2012), "Analyzing The MD5 Collision In Flame", Presentation at SummerCon.