JOURNAL OF COMPUTING TECHNOLOGIES

ISSN 2278 – 3814

Available online at www.jctjournals.com

Volume 3, Issue 1, July 2012

New Modified 256-bit Message Digest Algorithm Based on Existing algorithms

Pankaj Kumar Jain[#1],Ravindra Gupta[#2],Gajendra Singh[#3]

M.Tech(CSE)Scholer, dept of CSE,dept ofCSE,SSSIST,Sehore

[1] *pankajjain80@gmail.com*
[2]*ravindra_p80@rediffmail.com*
[3]*gajendrasingh86@rediffmail.com*

Abstract-
This paper describes the New MD algorithm base on previous algorithms with new chaining variable. It analyses the theories from program codes and sums up some current crack approaches of this algorithm. According to these crack ways, the paper brings forward the corresponding measures for improvement and adopts procedures to achieve a good algorithm to prove its validity.

Key Words: - Hash function; MD algorithm; Compressed function and Hash code length; Collision and Attack.

Introduction

With the increasing popularity of computers and the Internet in the past two decades, people have paid more and more attention on information and network security which results in a number of Encryption algorithms coming into being. These algorithms are currently the mainstream for the cryptographic check and file check. In the databases of many sites, even in the UNIX and LINUX operating systems, users login passwords to preserve by taking the check form of MD5 or SHA .However, as time goes by, the security of the algorithm is not as good as those years. Therefore, this paper puts forward a series of improvement program about MD5 algorithm to make its safety performance be improved.

The Brief Introduction of MD5

MD5, with the full name of the Message-digest Algorithm 5, is the fifth generation on behalf of the message digest algorithm. In August 1992, Ronald L.Rivest [3] submitted a document to the IETF (The Internet Engineering Task Force) entitled "The MD5 Message-Digest Algorithm", which describes the theory of this algorithm. For the publicity and security of algorithm, it has been widely used to verify data integrity in a variety of program languages since the 1990s.

Message Digest describes a mathematical function that can take place on a variable length string. The number 5 simply depicts that MD5 was the successor

to MD4 [2]. It can compress any length of data into an information digest of 128bits while this segment message digest often claims to be a digital fingerprint of the data. This algorithm makes use of a series of non-linear algorithm to do the circular operation, so that crackers cannot restore the original data. In cryptography, it is said that such algorithm as an irreversible algorithm, can effectively prevent data leakage caused by inverse operation. Both the theory and practice have good security, because the use of MD5 algorithm does not require the payment of any royalties, time, and cost less which make it be widely used in the general non-top-secret applications. But even the top-secret area, MD5 may well be an excellent Intermediate technology.

MD5 is essentially a checksum that is used to validate the authenticity of a file or a string and this is one of its most common uses. Let's take a look at a working example. Let's say you have released some software or a program that you want people to freely distribute, this is all good and well but what if someone was to tamper with your application with malicious intent? For example what if they added malware onto your program, how would people know? Well if you had taken an MD5 checksum of your original program and made this information

public, then when people downloaded your software could then check their downloaded file and check that the MD5 checksum matches yours. If it does then great! If not then it means your program has been tampered with.

What's poor in MD5

MD5 is a famous cryptographic hash function example. It is widely used in Internet applications. It was published as the RFC-1321 Internet standard in 1992 [3]. It hashes arbitrary length bit strings onto 128 bits and uses the Merkle-Damgard construction from a $128 \times 512 \rightarrow 128$ compression function. The compression function is made from an "encryption function" by the Davies-Meyer scheme and will map a 128- bit value H = (A, B, C, D) and a 512-bit key block B into a 128-bit value. Actually 128-bit hash value is not adequately long to stop birthday attacks and two messages that have the same hash value could be found easily so MD5 is no longer secure, and it is not recommended for use in the future [4].

The Brief Introduction of SHA

SHA (Secure Hash Algorithm) is a 160-bit hash function published in 1993 as the Secure Hash Standard by NIST (The National Institute of Standards and Technology) [5]. It is based on MD5 and is mainly used in digital signature schemes. It hashes onto 160 bits and uses the Merkle- Damgard construction from a $160 \times 512 \rightarrow 160$ compression function. As for MD5, the compression functions of SHA and SHA-1 are made from an "encryption function" by the Davies-Meyer scheme.

A 160-bit hash function has a security level on the order of 80 bits, so SHA-1 is designed to match the security level that uses an 80-bit secret key [1]. SHA-0 analyzed by Chabaud and Joux using differential methods (local collisions and disturbance vectors) and they found a collision attack on SHA-0 of complexity 261 [11]. Biham and Chen found near collisions on SHA-0 in complexity 240 [12]. The work of Biham, Joux, and Chen included the first real collision of SHA-0 therefore; the migration to more secure hash functions should be accelerated. In 2001, NIST developed three new hash functions SHA-256, 384, and 512 whose hash value sizes are 256, 384, and 512 bits, respectively. These hash functions are standardized with SHA-1 as SHS (Secure Hash Standard) [6, 7], and a 224-bit hash function, SHA-224, based on SHA-256, was added to SHS in 2004 but moving to other members of the SHA family may not be a good long term solution [8].

What's poor in SHA

In the past few years, there have been significant research advances in the analysis of hash functions and it was shown that none of the hash algorithm is secure enough for critical purposes. As mentioned before SHA-0 changed to SHA-1 because SHA-0 differential paths had a problem (i.e. ΔA, ΔB, ΔC, ΔD, $\Delta E = 0$) in the middle of calculation and collision will find easily as shown in Table 3. Although SHA-1 tried to remove this flaw but the weakness still remained [9, 10]. In fact differential cryptanalysis works when the attacker can predict the evolution of differences with a high probability because of existence of Neutral Bits. It is easy to cancel a difference in the state by changing compress functions before starting next round or by creating another difference in the messages that is prepared in our algorithm but it is obvious that they should change by a procedure to keep the solidarity regarding to Merkle-Damgard theory which proved that if the compression function is collision-resistant, then the hash function is collision resistant as well and mathematical induction need solidarity [1].

The weakness in SHA family originated from this fact that possibility of two different input value will produce the same output value in the middle of algorithm and it is important to have a good diffusion so that the output in each round will be spread out and not to be equal with the same output in the next coming stages. This has done with XOR-ing each stage output with next input something that has already done in CBC/MAC then the difference of outputs will be ensured but the previous digest algorithms does not use this technique. At the same time Double-Davies-Meyer scheme will ensure the diffusion and will resist against hackers to reach the minimum distance. In this stage it is shown how it is possible to combine two methods to have a good result with multiple security level that has been discussed before.

Description of the New Message Digest Algorithm

This algorithm basically based on MD5 algorithm [3]. MD5 is a non-reversible encryption algorithm. It is widely applied in many aspects, including digital signature, encryption of information in a database and encryption of communication information. It makes large amounts of information to be compressed into a confidential format before signing the private key by digital signature soft (that is, any length byte string is transformed into a certain length of big integer).

A brief description of new modified MD algorithm as follows: MD algorithm divides plaintext input into blocks each which has 512-bit, and each block is again divided into sixteen 32-bit message words, after a series of processing, the outputs of the

algorithm consist of eight 32-bit message words. After these eight 32-bit message words are cascaded, the algorithm generates a 256-bit hash value which is the required cipher text. Specific steps are as follows.

(1) **Padding-bit:** Without loss of generality, supposes that the original data at the source has k bits (mk-1, mk-2… m0), where mi_ {0, 1}. For MD algorithm, its k bits data must be processed in 512-bit message block, so if the length of source is less than that length, padding is always added until its length in bits is congruent to 448 modulo 512 (length≡448 mod 512). The padding consists of a single 1-bit followed by the necessary number of 0-bits.

(2) **Appending the length of data:** A 64-bit representation of the length on bits of the original message is appended to the result of above step. It is present by two 32-bitdigits. At this time, the length of message is filled to a multiple of 512.

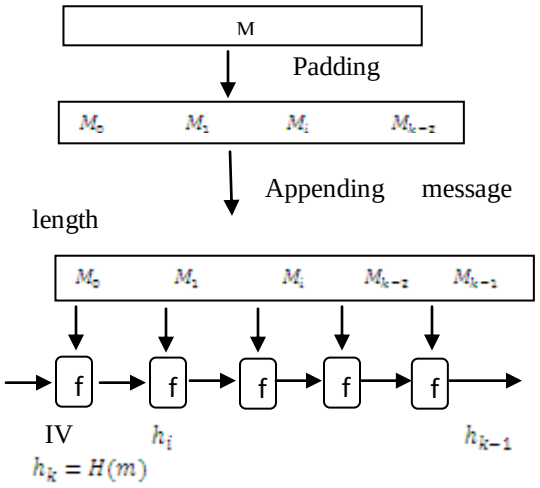

Figure 1: Working principle of an iterated hash function

(3) **Initialize MD Standard parameters:** eight 32-bit integers A, B,C,D,E,F,G,H are called chaining variables, used to calculate the message digest, are initialized by hexadecimal number

A=0x01234567
B=0x89abcdef
C=0xfedcba98
D=0x76543210
E=0x12ac2375
F=0x3b341042
G=0x5f62b97c
H=0x4ba763ed

Bit operation functions: We define four auxiliary operation functions J, K, L and M respectively, in which x, y, z, p, q, r, and s are seven 32-bit integers that each take as input 32-bit words and produce as output one 32-bit word.. The operation is as follows:

J(x,y,z,p,q,r,s) = (x∧y)∨((¬x)∧z)∨(p∧q)∨ ((¬p) ∧ r) ∧s ……..……1

K(x,y,z,p,q,r,s) = (x∧ z) ∨(y∧ (¬z)) ∨ (p∧ r) ∨(q∧ (¬r)) ∧s ……....2

L(x,y,z,p,q,r,s) = (x⊕y⊕z)∨(p⊕q⊕r) ∧s ……..3

M(x, y, z,p,q,r,s) = y ⊕ (x ∨ (¬z)) ∨q ⊕ (p ∨ (¬r)) ∧s ……..…..…..4

In four functions, if the corresponding bits of x, y z, p, q r and s are independent and uniform, then each bit of the results should be independent and uniform as well. For

$$x = \sum_{i=1}^{32} x_i\, 2^{i-1} \in Z/(2^{32}),\ x^i \in \{0,1\}$$

We call x^i the i-th bit of x.

(4) **Main transformation process:** The number of main looping this algorithm is the number of 512-bit information groups. The main loop have four rounds, each round carries out 16 operations, so the total of operations are 64 steps. The above eight chaining variables are assigned to another eight chaining values: a0=A, b0=B,c0=C, d0=D, e0=E, f0=F, g0=G, h0=H. One of the chaining values is updated in each step and computation is continued in sequence. Here we have defined four rounds composite functions of main loop FF, GG, HH and II respectively. The operation is as follows:

FF →a = b + ((a + J (b c d e f g h) +M_i+ t_i)<< s)……...5

GG →a = b + ((a + K (b c d e f g h) +M_i+t_i)<<s)……..6

HH →a = b + ((a + L (b c d e f g h) +M_i+t_i)<<s)……..7

II →a = b + ((a + M (b c d e f g h) +M_i+t_i) <<s)……8

Where, + is addition modulo2^{32}, Mi (0_i_15) is a 32-bit message word and the 512-bit message block is divided into 16 32-bit message words. x _ s is the left

shift rotation of x by s bits. The ti and s are step-dependent constants, ti has the following options: in i-th step, ti is the integer part of 4294967296×abs (sin (i)),4294967296=2^{32}.

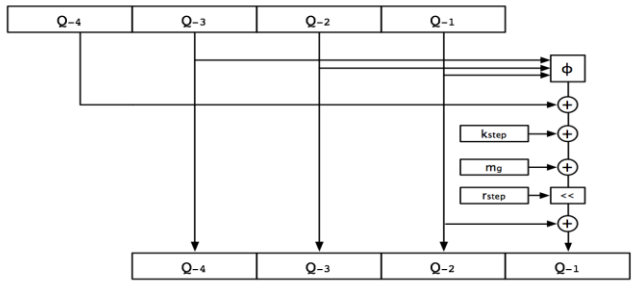

Figure 2: Standard step of compression function in MD5

After all of these steps, A, B, C, D, E, F, G, H add a, b, c, d, e, f, g, h Respectively, then the algorithm is continued to run the next 512-bit message block, the final output is A, B, C, D, E, F, G, H of cascading. Application of MD algorithm is to generate a message digest of information in order to prevent tampering. We view the entire file as a large text message, and result in a unique message digest by the irreversible string transform method. In the future, if the contents of file are changed, we only recalculate message digest of this file, and will find the difference from the original message digest. There by, we can sure the checked file is incorrect.

Attacks on MD Algorithms:

Like every cryptographic function, hashes are susceptible to brute-force attacks. The longer L is, the more work an attacker has to do to mount an attack; however, hashes with longer L also are usually slower to computer. There are three important attacks on hashes:

1. A "collision attack" allows an attacker to find two messages M1 and M2 that have the same hash value in fewer than 2^(L/2) attempts.
2. A "first-preimage attack" allows an attacker who knows a desired hash value to find a message that results in that value in fewer than 2^L attempts.
3. A "second-preimage attack" allows an attacker who has a desired message M1 to find another message M2 that has the same hash value in fewer than 2^L attempts.

1. Brute force attack:

In cryptography,a brute force attack or exhaustive key search is a strategy that can in theory be used against any encrypted data[13] by an attacker who is unable to take advantage of any weakness in an encryption system that would otherwise make his/her task easier. It involves systematically checking all possible keys until the correct key is found. In the worst case, this would involve traversing the entire search space.

Symmetric key length vs brute force combinations

Key size in bits[2]	Permutations	Brute force time for a device checking 256 permutations per second
8	28	0 milliseconds
40	240	0.015 milliseconds
56	256	1 second
64	264	4 minutes 16 seconds
128	2128	149,745,258,842,898 years
256	2256	50,955,671,114,250,100,000,000,000,000,000,000,000,000,000,000,000,000,000,000 years

Table 1 Combinations Time

2. Rainbow tables:

A rainbow table is a precomputed table for reversing cryptographic hash functions, usually for cracking password hashes. Tables are usually used in recovering the plaintext password, up to a certain length consisting of a limited set of characters. It is a form of time-memory tradeoff, using less CPU at the cost of more storage. Proper key derivation functions employ salt to make this attack infeasible. Rainbow tables are a refinement of an earlier, simpler algorithm by Martin Hellman[14] that used the inversion of hashes by looking up recomputed hash chains

3. Birthday attack:

A Birth day attack is a name used to refer to a class of brute-force attacks. It gets its name from the surprising result that the probability that two or more people in a group of 23 share the same birthday is greater than 1/2; such a result is called a birthday paradox. If some function, when supplied with a random input, returns one of k equally-likely values, then by repeatedly evaluating the function for different inputs, we expect to obtain the same output after about 1.2k1/2. For the above birthday paradox, replace k with 365. Birthday attacks are often used to find collisions of hash functions

Results and Discussions

All the attacks of Joux et al., Biham & Chen [9], and Wang et al.[4] are differential and they use this fact

that by changing a small number of message bits, it is possible to cancel the difference after a few rounds, or keep the Hamming distance low. Differences are usually defined as the XOR of the value in one run and the corresponding value in the other run (or alternatively additive or multiplicative difference)

We have presented a new hash function based on Double-Davies-Meyer that satisfied Merkle-Damgard condition. Security of this new algorithm is higher than SHA-1 and MD5 because even if local collision happened in the middle of SHA second algorithm will fade it with an acceptable diffusion so at the start of next round there are no equal states with previous round. It means that Hi – Hi-1 ≠ 0 We chose some messages that has already shown as collision in MD5 and SHA-1 and changed them by XOR and Addition but we did not find any collision. Sophisticated message modification techniques were applied to achieve the necessary conditions on the chaining variables and the message bits. The differential path for the improved algorithm is different from the previous differential path so it is resistant against local collision and differential attack.

In this scheme brute force attack takes 256 Permutations to break algorithm which takes more time than 160 bit hash function as show in the Table1. Even the last scheme is 160 bits and need 280 bit for birthday paradox but it is strong enough to the first and second preimage attack. We can extend the length of hash to 512 or 640 to be more resistible against birthday attack which comparing with its ancestors, it is more powerful.

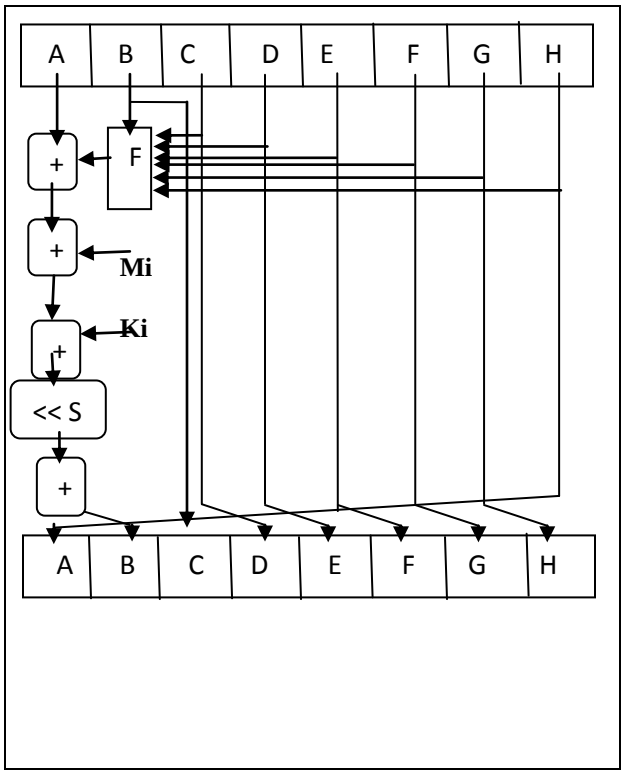

Fig.3 Diagram of the algorithm

Conclusion and Future work

Recently, a pure MD5 encryption still has been widely used. However, with the rapid development of CPU technology, the crack speed will be faster and faster. We might welcome a DIY (do it yourself) era of MD5 class HASH algorithm after a network security crisis in the future. In this paper proposed a new message digest algorithm based on previous algorithms that can be used in any message integrity or signing applications. Most of cryptanalysis tries to find collision based on Differential attack but there is no way to find neutral bits for this kind of attack in parallel scheme.

We can extend our algorithm to have a bigger size of hash (512, 768 …) like SHAs by extending the block size of compression functions or increasing number of them.

Reference

[1]. S.Vaudenay "A Classical Introduction to Cryptography Applications for Communications Security" Springer, 2006, P 74.

[2] R. L. Rivest, "The MD4 message digest algorithm." Advances in Cryptology - CRYPTO ' 90, vol. LNCS 537, pp. 303 -311, 1991.

[3] R. Rivest. The MD5 Message-Digest Algorithm [rfc1321]

[4]. X. Wang, X. D. Feng, X. Lai and H. Yu., "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," Cryptology ePrint Archive: Report 2004/199, Aug. 2004 http://eprint.iacr.org/2004/199/

[5]. NIST, "Secure Hash Standard," FIPS PUB 180, May. 1993.

[6]. NIST FIPS PUB 180-1. Oct.2001.

[7]. NIST, "Secure Hash Standard (SHS)", FIPS PUB 180-2, 2002.

[8]. S. Chang, M. Dworkin, Workshop Report, The First Cryptographic Hash Workshop, Report prepared, NIST 2005.

[9]. E. Biham, R. Chen, "New results on SHA-0 and SHA-1" Crypto 2004 Rump Session, Aug. 2004.

[10]. K. Matusiewicz and J. Pieprzyk "Finding good differential patterns for attacks on SHA-1" eprint 2004 Available : http://eprint.iacr.org/2004/364.pdf.

[11]. F. Chabaud, A. Joux. "Differential Collisions in SHA-0". In Advances in Cryptology CRYPTO'98, Santa Barbara, CA, Lecture Notes in Computer Science 1462. Springer-Verlag, NY, pp. 56–71, 1998.

[12]. E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby and C. Lemuet. "Collisions in SHA-0 and Reduced SHA-1- In Advances in Cryptology" – Eurocrypt'05, Springer-Verlag, 2005.

[13]ChristofPaar, Jan Pelzl, Bart Preneel (2010). Understanding Cryptography: A Textbook for Students and ractitioners.Springer.p. 7. ISBN 3642041000.

[14]M.E. Hellman, H.R. Amirazizi, "A Cryptanalytic Time - Memory Trade-Off," IEEE Transactions on Information Theory, vol. 34-3, pp. 505-512, 1988

Message	MD5 (128bits)	SHA-1 (160bits)	NEW MD (256bits)
""	D41D8CD98F00B204E9800998ECF8427E	DA39A3EE5E6B4B0D3255BFEF95601890AFD80709	818872413013d2cfb89c56a28977d237d0f0f41a3c4afde4f83480a0f9026695
"a"	0CC175B9C0F1B6A831C399E269772661	86F7E437FAA5A7FCE15D1DDCB9EAEAEA377667B8	2f983d73f9e3f8e132aaf074b8d4d13b53d88a637b97c2c2871b9df969a0bf5f
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789"	D174AB98D277D9F5A5611C2C9F419D9F	761C457BF73B14D27E9E9265C46F4B4DDA11F940	2c8c3dd50958f3d8f1d477c7b7eebfad8bae094dc97691bf604df5a7693d21cb

Table. 2 Result

Message 1	A766A602 B65CFFE7 73BCF258 26B322B3 D01B1A97 2684EF53 3E3B4B7F 53FE3762 24C08E47 E959B2BC 3B519880 B9286568 247D110F 70F5C5E2 B4590CA3 F55F52FE EFFD4C8F E68DE835 329E603C C51E7F02 545410D1 671D108D F5A4000D CF20A439 4949D72C D14FBB03 45CF3A29 5DCDA89F 998F8755 2C9A58B1 BDC38483 5E477185 F96E68BE BB0025D2 D2B69EDF 21724198 F688B41D EB9B4913 FBE696B5 457AB399 21E1D759 1F89DE84 57E8613C 6C9E3B24 2879D4D8 783B2D9C A9935EA5 26A729C0 6EDFC501 37E69330 BE976012 CC5DFE1C 14C4C68B

	D1DB3ECB 24438A59 A09B5DB4 35563E0D 8BDF572F 77B53065 CEF31F32 DC9DBAA0 4146261E 9994BD5C D0758E3D
Message 2	A766A602 B65CFFE7 73BCF258 26B322B1 D01B1AD7 2684EF51 BE3B4B7F D3FE3762 A4C08E45 E959B2FC 3B519880 39286528 A47D110D 70F5C5E0 34590CE3 755F52FC 6FFD4C8D 668DE875 329E603E 451E7F02 D45410D1 E71D108D F5A4000D CF20A439 4949D72C D14FBB01 45CF3A69 5DCDA89D 198F8755 AC9A58B1 3DC38481 5E4771C5 796E68FE BB0025D0 52B69EDD A17241D8 7688B41F 6B9B4911 7BE696F5 C57AB399 A1E1D719 9F89DE86 57E8613C EC9E3B26 A879D498 783B2D9E 29935EA7 A6A72980 6EDFC503 37E69330 3E976010 4C5DFE5C 14C4C689 51DB3ECB A4438A59 209B5DB4 35563E0D 8BDF572F 77B53065 CEF31F30 DC9DBAE0 4146261C 1994BD5C 50758E3D
Hash Output	C9F16077 7D4086FE 8095FBA5 8B7E20C2 28A4006B

Table. 3 SHA-0 Collision Detected