Integrtation of Encryption and Hash Function for Improved Message Authenticity

Richa Purohit (Arya), Yogendra Singh, Dr. Upendra Mishra, Dr. Abhay Bansal
Amity School of Engineering. & Technology Amity University Rajasthan Jaipur, India
Dy. Manager- IT, Au Financiers India Pvt. Ltd. Jaipur, India
Amity School of Engineering. & Technology Amity University Rajasthan Jaipur, India
Amity School of Engineering. & Technology Amity University Uttar Pradesh Noida, India

Abstract

Hash function is an important tool for implementing data and information integrity. Presently there are many integrity techniques, that support hashing, but almost every technique faces one or other attack or any other security or performance related issue. The main problem is the possibility of creating forged hash value by intruder, which may be transferred with the changed message, and being received and accepted by receiver. This paper discusses how to provide data origin authenticity along with data integrity by integrating symmetric key encryption algorithm with a hash algorithm. Hash functions provide data integrity, while encryption techniques provide source or origin authenticity by using a shared secret key. In the paper the technique to combine both hash function and encryption algorithm-DES is given so that, both features, data integrity and source authentication, may be availed while communicating message between sender and receiver on a network.

Keywords: DES, hash function, MD5, message authentication, message integrity

I. INTRODUCTION

Hash functions are one of the fundamental importances in cryptographic protocols. They are applied in digital signatures, data integrity, time stamping, password verification, digital watermarking, group signature, e-vash and in many other cryptographic protocols.

Hash Functions take a block of data as input, and produce a hash or message digest as output. The usual intent is that the hash can act as a signature for the original data, without revealing its contents. Therefore, it's important that the hash function be irreversible - not only should it be nearly impossible to retrieve the original data, it must also be unfeasible to construct a data block that matches some given hash value. A hash function takes a long string (or message) of any length as input and produces a fixed length string as output, sometimes termed a message digest or a digital fingerprint[1].

II. CLASSIFICATION OF HASH FUNCTIONS

For data origin authentication there is a special class of hash functions that use a key. The hash functions without a key are used for data integrity.
According to [2] a function used mainly to detect changes in the signed messages is called modification detection code (MDC) or manipulation detection code, and less commonly as message integrity code (MIC). MDC is a subclass of unkeyed hash functions.

A one-way hash function (OWHF) is MDC for which it is difficult to find an input which hashes to a prespecified hash-value.
A collision resistant hash function (CRHF) is characterized by difficulty in finding any two inputs having the same hash-value.

For data origin authentication purpose message authentication codes (MAC) are used. The purpose of a MAC is to facilitate, without the use of any additional mechanisms, assurances regarding both the source of a message and its integrity. MACs have two functionally distinct parameters, a message input and a secret key.
MACs are keyed hash functions [2]. In case of MAC, the design intent is to be infeasible to produce the same output without knowledge of the key.

Currently MD5 [3] and SHA-1 [4] are widely used all over the world as estab;lished hash functions. Both of these hash functions are derived from MD4 [5]. Successful attacks have been performed on MD4, so all hash functions that are based upon its structure may also have common weaknesses.

In this paper, we will describe the design algorithm of cryptographic hash function along with the use of Data Encryption Standard (DES). Here, DES is an already proven symmetric encryption scheme that produces cipher text in 64 bit block taking 64 bit plain text as input. Merging of encryption algorithm will provide additional facility of source authentication and thus, will improve stucture of hash function that supports message integrity only. Here authentication means to provide a means to receiver for assurance that the message is actually sent by original sender, not by some intruder, and at the same time the message is also authenticated and original. The paper also discusses the working of MD5 and DES functions and there key features. Then it describes the proposed method and analyzes its cost and potential applications.

III. PROPERTIES OF HASH ALGORITHM

Some of the properties of hash functions are due to the requirements in implementation. For instance, it is useful to have a hash function which is easy to implement (easy to compute the hash of a message) on one side and on the other side it has to be able to compress the information (the message). Other properties are driven from the cryptographic environment requirements. As such we have three properties[6]:

* Preimage Resistance (one way function) – Given a hash h it should be difficult to find any message **m** such that h= hash(m). This concept is known as one-way function. Functions that lack this property are vulnerable to preimage attacks.

* 2nd-Preimage Resistance (also known as collision resistance) – Given an input **m1**, it should be difficult to find another input **m2** where m1 ≠ m2 such that hash(m1) = hash(m2). This property is sometimes known as weak collision resistance, and functions that lack this property are vulnerable to second-preimage attacks.

* Collision Resistance (also called strong collision resistance) – It should be difficult to find two different messages m1 and m2 such that hash(m1) = hash(m2). Such a pair is called a cryptographic hash collision. This property is sometimes known as strong collision resistance. It requires a hash value at least twice as long as that required for preimage-resistance, otherwise collisions may be found by a birthday attack.

These properties imply that a malicious adversary cannot replace or modify the input data without changing its digest. Thus, if two strings have the same digest, one can be very confident that they are identical.

The one-way hash function is a hash function (i.e., offering ease of computation and compression) with the additional properties, as defined above: preimage resistance, 2nd-preimage resistance [7]. The collision resistant hash function is a hash function characterized by 2nd-preimage resistance and collision resistance.

In our solution, we will use MD5 algorithms. The MD5 is a widely used algorithm to verify data integrity through the creation of a 128-bit message digest from data input (which may be a message of any arbitrary length) that is claimed to be as unique to that specific data as a fingerprint is to the specific individual. MD5, which was developed by Professor Ronald L. Rivest of MIT, is intended for use with digital signature applications, which require that large files must be compressed by a secure method before being encrypted with a secret key, under a public key cryptosystem. MD5 is currently a standard, Internet Engineering Task Force (IETF) Request for Comments (RFC) 1321. According to the standard, it is "computationally infeasible" that any two messages that have been input to the MD5 algorithm could have as the output the same message digest, or that a false message could be created through apprehension of the message digest. MD5 is the third message digest algorithm created by Rivest. All three (the others are MD2 and MD4) have similar structures, but MD2 was optimized for 8-bit machines, in comparison with the two later formulas, which are optimized for 32-bit machines. The MD5 algorithm is an extension of MD4, which the critical review found to be fast, but possibly not absolutely secure. In comparison, MD5 is not quite as fast as the MD4 algorithm, but offers much more assurance of data security.

Following are the steps involved in MD5 algorithm to create a digest value:

1. First of all the original message is padded with 100…00 bits, so that the original message length ≡ 448 mod 512.

2. As next step, original message length (in 2^{64} bit representation) is appended to the output of previous step.

3. After initialization of 128 bit MD buffer, the message (output from previous stage) is processed in blocks of 512 bit each. Processing is done on individual blocks, where, each step consists of 4 individual rounds, and each round contains 16 steps, thus total 64 steps.

4. After processing each individual 512 bit block, the output is taken in the 128 bit buffer, and this buffer is used as current value in processing of next 512 bit block. In such a way, after processing all 512 bit blocks, the final output in buffer, obtained by processing last block of message, is termed as final message digest value of the whole message.

So, If we have two distinct messages, M1 and M2, the difficulty of computing their digest, such that MD5(M1) = MD5(M2), is in the order of 2^{64} operations. Similarly, for a given a message digest h, the difficulty of computing a message, M such that MD5(M) = h, is on the order of 2^{128} operations.

An attack on MD5 was presented in 2005, using differential analysis, which allows finding collisions efficiently[8]. The same attack, applied on HAVAL-128, MD4, RIPEMD, and SHA-0 reduced the number of operation for determining a second message with the same hash. Even if the number of operations required for the attack is considerable, such attacks are reducing the ideal number of operations assumed to be required for breaking hash functions. Such findings motivated NIST to find new, resistant hash functions [9].

IV. Properties of The DES Block cipher

DES originated at IBM in 1977 and was adopted by the U.S. Department of Defense. It is specified in the ANSI X3.92 and X3.106 standards and in the Federal FIPS 46 and 81 standards. DES is a block cipher encryption algorithm i.e. it takes input in block and produces output also in block. The block size is specific for each algorithm. DES deals with 64 bit block [10]. Cryptographers say that a block cipher is secure if both $C \rightarrow E(P)$ and $P \rightarrow D(C)$ are indistinguisable from a randomly selected permutation. This cascade construction extends the collision resistance and pre-image resistance[11], (here, C= Cipher text, E= Encryption algorithm, P= Plain text and D= Decryption algorithm). DES is symmetric encryption technique, that is it uses similar key for both encryption and decryption algorithms.

In processing, DES consists of two permutation operations with 16 identical rounds of operations in between. It uses a 64 bit long key, where 8 bits are reserved as parity bits (one bit for each of the eight words in the key), thus effective key length is 56 bit. For each of the sixteen round a 48 bit key is used, which is made up by permutation, combination, shifting and other operations performed on initial 56 bit key, and all 16 keys are diferent each time. The performance of a block cipher is dependent on the cost of both the encryption routine and key setup.For bulk encryption the cost of a single key setup is amortized over the entire encryption session. However, when used as the basis for a hash function, the cost of the key schedule becomes a significant factor. Most modern ciphers, including the DES, tend to have a lightweight key schedule[12].

V. Proposed Solution

One approach would be to try to buid the new function based on existing techniques only. For example- we may concatenate the outputs of two different hash functio techniques and treat this concatenated output as final digest. But for this solution, both of the existing hash functions should be independent of each other.

One more method may include the techniques to strengthen the existing hash algorithms by inceasing the number of rounds, adding some coding or scrambling steps, increasing the buffer size (so as to increasing the digest size), making the mixing step varing with the rounds[13].
Our method involves combination of two established techniques. Now, we describe our newly proposed hash function based on MD5 and DES. The hash function has following properties-
1. It produces a 128 bit digest value. $\{0,1\} < 2^{64} \rightarrow$ 128 bit value. Here $\{0,1\} < 2^{64}$ denotes the set of all messages whose length is at most 2^{64}-1 which is reasonable in all practical applications.
2. Our hash function is also a wide pipe hah function. Like other hash functions we will use an initial value and a variant of padding rule which provides a dynamic hash.
Algorithm:
1. Padding - Padding is done in following two steps-
a. Pad1:-In the first step, the given message is padded so that $|M| < 2^{64}$. The padding is done by 1 followed by neccasray number of 0 bits:-
$\text{Pad1}(M) = M||1||0^k$
b. Pad2:- (Append message length) In the second step, the output of step (a) is padded with 64 bit binary representation of message length. i.e. pad2 $(M) = M||1||0^k||\text{bin}_{64}(|M|)$.
2. Generate intermediate hash- let $M_1|| M_2|| ---||M_t$ be the padded message and each M_i is a 512 bit block of message. We initialize an MD buffer with some predefined initial values, i.e. $(S_0, j_0) = (S^{IV}, 0)$. Now, we invoke a SHA-1 like compression function C and produce a 128 bit intermediate hash value.
$(S_0, j_0) \xrightarrow{M1} (S_1, j_1)$
3. Apply Symmetric Encryption Algorithm- After getting intermediate digest value; we divide it into two blocks of 64 bit each. And apply symmetric key encryption algoroithm on each block separately, which again gives us two individual blocks of 64 bits as output.
4. Getting output of processing of a single block- the outputs of encryption on two 64 bit blocks are combined together (by simple append operation) and one 128 bit block is formed. (After first block it is known as (S_1, j_1) and so on.) (Figure 1).
5. Output:- This new (Sn, jn) is used as input values for IV for next 512 bit block of message for processing. After processing all t blocks, the final (S_t, j_t) is final hash value for initial message. (Figure 2)

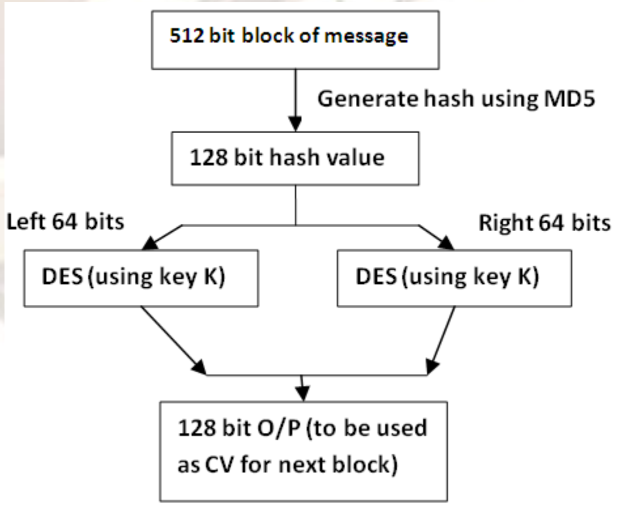

Figure 1:Intermediate processing of a single 512 bit block

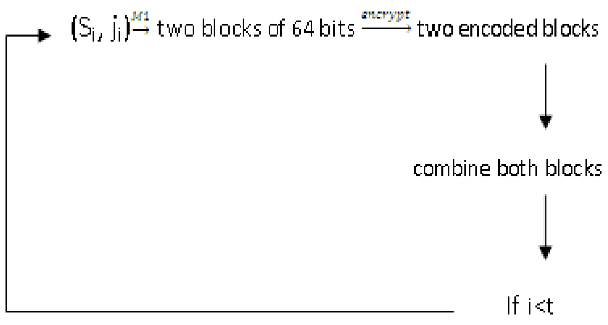
$(S_i, j_i)\xrightarrow{M1}$ two blocks of 64 bits $\xrightarrow{encrypt}$ two encoded blocks
combine both blocks
If i<t

Figure 2:Final Procesing

Use of DES as an intermediate symmetric encryption algorithm needs Key Distribution Center (KDC) [14], that will distribute the symmetric key (or session key) to both parties- sender and receiver in encrypted form, using its own private key, so that no other party in the network may gain access to the key and the proposed solution is secure in this manner.

The proposed solution increases the present processing of a single block, as does not directly feeds the output of previos block as current value for next block, rather it first devides the 128 bit output into two 64 bit partitions, encrypts them using DES algorithms, and further uses the output after concatednation of both 6 bit cipher blocks as current value for next block.

CV0=IV

CVq= E (K,B1) || E (K,B2)

Where,

IV= Initialization value of MD buffer set by MD5

E= DES scheme

B1= Left 64 bits of output of MD5 digest value

B2= Right 64 bits of output of MD5 digest value

K= DES key

Following diagram shows overall processing of the algorithm.

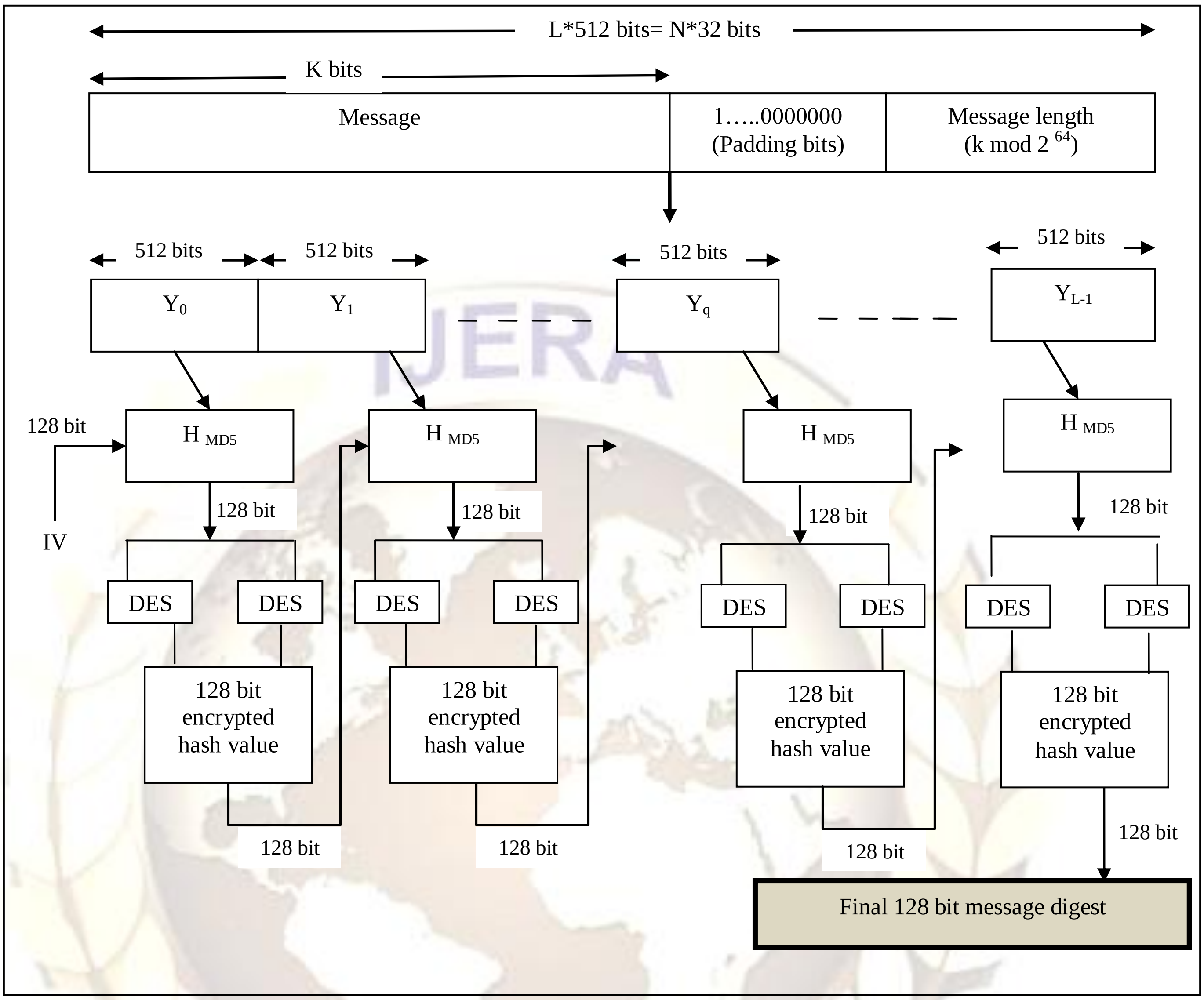
L*512 bits= N*32 bits
K bits
Message
1…..0000000 (Padding bits)
Message length (k mod 2 64)
512 bits
512 bits
512 bits
512 bits
Y0
Y1
Yq
YL-1
H MD5
128 bit
IV
DES
128 bit encrypted hash value
128 bit
Final 128 bit message digest

Figure 3: An illustrated view of processing of MD5 having DES in between

VI. VII Analysis of Proposed Method

One major drawback of using DES algorithm while generating the hash is the comparative slow speed of DES, which may ultimately slow down the whole process. This is because the usual DES Encryption algorithm streches the given 56-bit DES key into 48*16 bits (16 rounds of operations, where each round uses different 48 bit key made up from initial 56 bit DES key). One way to improve the rate of DES based hash algorithm would be to skip the key-scheduling algorithm and feed 16*48 bits of input text directly as a key. This consuiderably increases the rate of digest construction. Using the streched output as the DES key would effectively allow us to compress 512 bits per DES-call.

The strength of the algorithm may taken by considering brute force attack on it. Here, the adversary may need to perform at least 2^{196} trials for a successful attack, that is equivalent to both attack on DES alone and attack on MD5 alone. It says that the algorithm is stronger than MD5. At the same time the cost of proposed solution is also more than simple MD5 as it combines two algorithms. But one Still the analysis of the performance of cryptographic algorithms is closely related to their security: high performance applications require an optimal trade-off between security and speed. [15]

The security of the proposed solution can be split into a consideration of underlying block cipher and then of the compression function and chaining mode. The latter concerns are handled by the results of Damgård, Merkle, Black et al [16], and Biham and Dunkelman [17] so for reasons of space we concentrate on the cipher within the compression function and particularly on DES.

VII. CONCLUSION

The poposed method is easy to implement and analyze, whih enhancec its acceptability and applicability. The algorithm is of importance where source authentication is equally important as that of message integrity. The solution considers two already established and world wide used two algorithms- MD5 and DES. The other similar solutions may also be provided combining any other hash function and encryption techiques such as SHA-1, whirlpool, Tiger etc or Advanced Encryption Standard (AES).

BIBLIOGRAPHY

[1] S. M. Bellovin and E. K. Rescorla. Deploying a new hash algorithm. In Proceedings of NDSS '06, 2006.

[2] P. Rogaway , T. Shrimpton. — Cryptographic Hash - Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. Springer-Verlag 2004.

[3] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321, April 1992.

[4] National Institute of Standards and Technology, U.S. Department of Commerce. Secure Hash Standard, 2002. FIPS PUB 180-2.

[5] R.L. Rivest. MD4 Message Digest Algorithm. RFC 1186, October 1990.

[6] M. Stanek. Analysis of Fast Block Cipher Based Hash Function. Computational Science and its Applications. 2006, vol-3982/2006, pp- 426-435. DOI: 10.1007/11751595_46

[7] R. Rivest, The MD4 Message Digest Algorithm, Procesedings of CRYPTO'90, August 1990.

[8] X. Wang and H. Yu, *How to Break MD5 and Other Hash Functions*, Advances in Cryptology – EUROCRYPT 2005, 2005, pages 19-35.

[9] R. Tirtea, *Cryptographic hash functions. Trends and challenges*, Journal of Computer Science and Control Systems , 2009, Vol 2, issue 2, pp- 62-65.

[10] M.E. Smid, D.K.Branstad. The Data Encryption Standard- Past and Future. Proceedings of IEEE, Vol 76, no.5, pages 42-64.

[11] J. Walker, M. Kounavis, S.Gueron, G.Graunke, *Recent Contribution to Cryptographic Hash Function*, Intel Technology Journal, 2009, Vol 13, issue 2, pp 80-95.

[12] B. Olivier, J.B.R. Mathew, S. Yannick, L.Y.Yiqun. *Looking Back at a New Hash Function*, ACISP 2008, LNCS 5107, pp. 239-253.

[13] S. Al- Kuwari, J.H Davenport, R.J. Bradford, Cryptographic hash functions: recent design trends and security notions, Short Paper Proceedings of 6th China International Conference on Information Security and Cryptology (Inscrypt '10). 2010, Science Press of China, pp. 133-150.

[14] L. Harn and C. Lin, Authenticated Group Key Transfer Protocol Based on Secret Sharing. Computers, IEEE Transactions on, June 2010, vol 59, issue- 3, pp-842-846.

[15] B. Preneel, *Software Performance of Encryption Algorithms and Hash Algorithms*, Appeared in Selected Areas in Cryptography, 2nd Annual International Workshop, SAC 1995, pp. 89–98, 1995.

[16] J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In M. Yung, editor, Advances in Cryptology – CRYPTO 2002, volume 2442 of Lecture Notes in Computer Science, pages 320–335. Springer-Verlag, 2002.

[17] E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions - HAIFA. Presented at Second NIST Cryptographic Hash Workshop, August 24-25, 2006. Available at: csrc. nist. gov/ groups/ ST/ hash/.