International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 6 (July 2014) http://ijirae.com

Implementation of New Modified MD5-512 bit Algorithm for Cryptography

Priyanka Walia
Student, M.Tech. CSE
GNDEC, Ludhiana

Vivek Thapar
Asstt. Prof., Deptt. of Computer Science & Engg.
GNDEC, Ludhiana

Abstract—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 whether it is MD5 or SHA-1.Nowadays scientists have found weaknesses in a number of hash functions, including MD5, SHA and RIPEMD so the purpose of this paper is combination of some function to reinforce these functions and also increasing hash code length up to 512 that makes stronger algorithm against collision attests.

Keywords— SHA, MD4, SCTP, Hash, MD5

I. INTRODUCTION

Hash algorithms are important components in many cryptographic applications and security protocol suites. In this paper, a unified architecture for MD5 and MD5-512 hash algorithms is developed. These two algorithms are different in speed and security level. Therefore, a unified hardware design allows applications to switch from one algorithm to another based on different requirements.

A hash function computes a fixed length output called the message digest from an input message of various lengths. The MD5 message digest algorithm, developed by Ron Rivest at MIT, accepts a message input of various lengths and produces a 128-bit hash code. It has been one of the most widely-used hash algorithms. However, it has been indicated in that there is a security threat in the algorithm. Furthermore, a 128-bit hash output may not offer sufficient security protection in the near future. To provide higher security protection, MD5-512 has been proposed by Dobbertin, Bosselaers and Preneel. MD5-512 accepts the same input format as that ofMD5, and produces a 512-bit output. It is easy to find that the structures of the two algorithms are quite similar. It follows that they can be combined together to give one hardware design that can perform the two hash functions. This approach has the following advantages. Firstly, the unified design is a resource-efficient implementation when different hash algorithms are needed in applications. Applications can switch to either algorithm based on different requirements. Second, since MD5 is still the most widely-used hash algorithm, upgrading the current implementation in the future to MD5-512 is much easier with unified hardware architecture.

Motivated by these observations, in this paper we develop a unified architecture for MD5 and MD5-512. Comparison with other unified architectures indicates that the proposed architecture is area-efficient. First, the input message is padded and divided into data blocks of length 512 bits. Each data block is treated as 16 32-bit words. The algorithms iteratively processes each data block. For the first data block, an initial value is used to compute an intermediate result. The intermediate result, called the chaining variable, is then updated according to the input data block and the previous result. After all iterations are done, the final chaining variable is the hash value.

Network Security & Cryptography is a concept to protect network and data transmission over wireless network. Data Security is the main aspect of secure data transmission over unreliable network. Data Security is a challenging issue of data communications today that touches many areas including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The rapid development in information technology, the secure transmission of confidential data herewith gets a great deal of attention. The conventional methods of encryption can only maintain the data security. The information could be accessed by the unauthorized user for malicious purpose. Therefore, it is necessary to apply effective encryption/decryption methods to enhance data security.

In the single phase of multiphase encryption is described as multiple encryptions where at each cycle different encryption key is used. In this encryption technique, decryption will be performed in reverse order. In multiphase encryption, such processes will be repeated number of times to enhance the complexity in encryption/decryption as well as security of data. Cryptographic algorithms and key sizes have been selected for consistency and to ensure adequate cryptographic strength for Personal Identity Verification (PIV) applications. Multiphase encryption may reduce the problem of key management in the existing technology of Personal Identity Verification (PIV) due to use of different encryption algorithms with fixed size keys instead of large number of variable length key. An Application of MD5 algorithm is implemented for the stream controlled transfer messages in the network. This would be a high security algorithm for data transfer in mobile networking with stream controlled logic. There may a vast number of applications for this algorithm in data transfer in various types of networks.

A. MD5 ALGORITHM

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 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.

MD5 was developed from MD, MD2, MD3 and MD4. 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 can not 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 an irreversible transformation transforming a set of data of any length into a hash value of 128-bit length and it is a consecutive processing method. Before operation, it first fills data to be processed, and adds 64-bit binary digits to the end of data representing the bit length of the original data. After filling, the bit length of data which is being processed becomes a multiple of 512. Then the data are divided into groups of 512 bits and computations are performed on each group orderly. The input of the first group operation is a 128-bit initial value; the input of the next group operation is a 128-bit output of the previous group operation. The 128-bit output of the last group operation is the MD5 hash value of the whole data. The key of MD5 algorithm is to perform 4 rounds of hash operation on the data packets of 512 bits. The processing logic is shown in Fig. 1

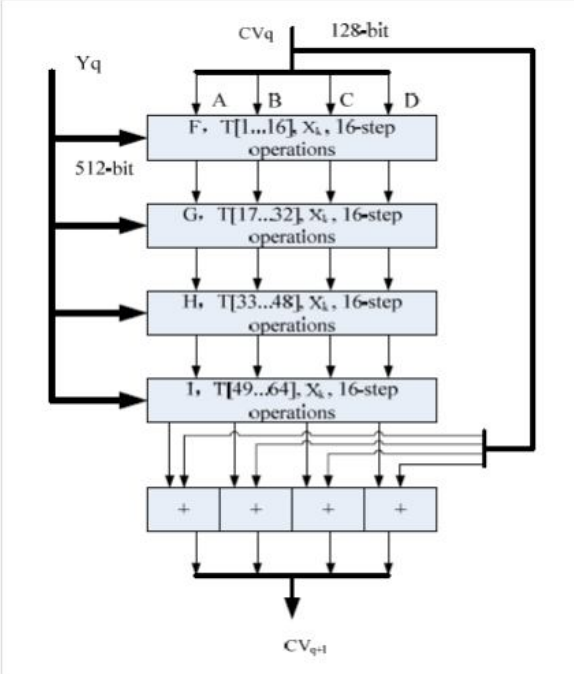

Figure 1: The processing logic

The 4-rounds of processing have similar structure but each of them has different logic function. The function used in each round is as follows.

F(x,y,z) = (x & y) | ((~x) & z) (1)
G(x,y,z) = (x & z) | (y & (~z)) (2)
H(x,y,z) = x ^ y ^ z (3)
I(x,y,z) = y ^ (x | (~z)) (4)

Here, we only briefly introduce the operation structure of MD5. Detailed information on MD5 can be found in RFC1321. In high-density data computing environment, the application of MD5 can be divided into two categories. One is to perform hash operation on a large data block and get its hash value, such as the ones used in data integrality validation. In this application, the data block requiring MD5 operation is larger, perhaps even reaching one million bytes or ten million bytes. The other is to perform hash operation on many small data blocks and gets a hash value of each small data block. Such as in the application of character string detection, we should detect a character string in a very large data dictionary,

in which input data must be divided into small pieces. In this application, each of the data blocks requiring MD5 operation is smaller, generally ranging from several bytes to hundreds of bytes. In order to distinguish the two categories, we call them large block hashing and large number hashing respectively.

B. MD5 Reinforcement

As we all know, if you want to crack passwords, it not only requires cipher text but also a cryptographic function. With the encryption function, you can derive the decryption function, let alone the MD5 is no inverse function. Even you can find the MD5 HASH by the 16-bit and 32 bit hex immediately, decryption is also very difficult because MD5 does not frequently exist by himself. And the most simple way is to reverse the original passwords or make ASCII yards plus N or conduct MD5 encoding together with certain prefixes and suffixes, which can be able to change the easy passwords into complex passwords, so that workers cannot easily get original passwords through MD5 when login in the websites or breaking. However, these methods have been adopted by many websites forums or software vendors.

i) The Reinforcement of Password Checkout

a) Simple Algorithm

With the reversed password, ASCII yards + N or the modified link variables is generally used in some security conscious sites or forums for the simple and effectiveness of algorithm process. They have excellent safety performance until the leak of encryption source codes.

b) Prefixes and Suffixes Algorithm

After encoding the MD5, you can save MD5 HASH and the original text of passwords with random characters together, and then conduct the comparison after adding this new version before and after the original text. The randomness is a major obstacle to decrypt, the workers can not be able to get the solution if they adopt the contemporarily popular law-breaking or query method. The famous FTP server software namely adopts this method.

c) The Secondary Encoding

The probability of cracking this method is zero after the worker continuing the secondary encoding that they encode the original password text plus MD5 encoding, because the primary text had been disordered after carrying out the first encoding .The famed Ten cent QQ also adopt this BASE64 + MD5 algorithm.

Such an approach is numerous, and you will be able to make it as long as you can think that. If you synchronously use these methods, MD5 check field will be able to revive in the password field.

ii) The Reinforcement of File Checkout

With regard to document checkout reinforcement, it is not so simple because the original file is equivalent to something that is proclaimed in writing. In the case of an express writing being published, the prefix method can be easily used to deceive the general computer users. The following methods can change experts' ideas of MD5 which is no longer suitable for the file checksum.

a) Key Document Method

The key document method is no longer a simple HASH checksum document, which will be added a small file of a few KB capacities. We tentatively named it as the key file. This method attaches key file to the original file to generate a new file, and then adopts MD5 encoding to the new file. The solution may be effective, but the check process will be more complex and the collision probability not has been diminished because the generated HASH still keeps the former length.

b) The Length Increase Method

Using this method to generate the HASH value may no longer be 128bit, but its multiples. This method is to open the original file in the form of 16 hexes and to carry out appropriate change to generate a new file at first, and then the worker makes a concatenation between the original file and the new MD5 HASH file to generate two or more times of the former HASH value's length. When the HASH becomes longer, its collision probability in theory, would become smaller. Therefore, I give a demonstration program in particular and divide it by 16 levels of security to prove my viewpoint.

This program uses the idea of extending HASH length to reduce the collision probability. It will open a file in the form of hex, and then put the figure 1 in the last digit of the file to get a new and different file whose number is decided by the security level (0 ~ 15).The workers calculate the MD5 HASH of the original files and these new files, then make a concatenation of these MD5 HASH to get a new string of long HASH value.

The program procedure:

a) Open the file read the file flux, after setting up its type and then shows it in the main window of the program.
b) Wait for user to choose level, the default is zero, which is also the original MD5code.
c) According to the level users choose, create new file by adding 1 to the previous file, and then conduct concatenation coding on the new file. During the process, the system will delete the new file automatically.

Thus,

- When the level set as 0, the procedure will carry through the primary MD5 coding.

- When the level set as 1, the system will make a concatenation the MD5 HASH with the new file created by adding 1 to the last number of the original file.
- When the level set as 2,the procedure will make a concatenation that created at level 1 with the MD5 HASH of the new life created by adding 2 to the last number of the original file.
- When the level set as 3,the procedure will make a concatenation that created at level 2 with the MD5 HASH of the new file created by adding 3 to the last number of the original file.

……

- When the level set as 15, the procedure will make a concatenation that created at level 14 with the MD5 HASH of the new file created by adding F to the last number of the original file.

This is just a simple and easy solution to implement the increasing length of HASH. In this set of proposals, the operation object of the procedure is not necessarily the last digit, because the first, the penultimate position, or a certain period of data streams can be suitable. In addition, the calculations do not necessarily be addition, multiplication, division, mixed operations or logic operation, as long as it on the basis of the original documents can be carried out.

Generated longer HASH by more complex computing can obtain more secure document verification.

d) If you want to directly compare with the foregone HASH strings and directly enter the HASH into the input text box to compare with the calculated result from the previous step. If you want to make a contrast between two files' HASH, click on [<<] or copy the generated HASH in the previous step into a textbox, and then open another file, select the same LEVEL, and calculate its HASH.

e) The program will compare the contents of the textbox on both sides, and reflect the comparison results to users through the message box,

This procedure generates a fourfold HASH than the length of the original MD5 HASH, Theoretically, reducing the probability of collision plus uncertainty of check levels; you will find it hard to create a crash file in response to it. I'm able to take the lead in putting forward this method and there are no similar tools for strengthening documents'MD5 checksum. After a little strengthening, MD5 becomes a rock solid. We can find that the algorithm is sufficiently tough vitality.

C. Application and Safety of MD5

i) Algorithm In Password Authentication

As the continuous development of computer network technology and rapid popularization of Internet application, ecommerce and e-government businesses are used more and more widely. However, at the time that these applications bring great convenience to our work and lives, they also bring increasingly serious security problems. Identity authentication is an important method to ensure safety of various e-commerce activities, e-government business information and internet information. Certificate based digital signature authentication and password authentication are most common at present. But traditional password authentication with basic mode of user name and password authentication faces many security problems, which concluded like these below:

1) Eavesdrop through user's host. There are various Trojan horse programs coming from different channels nowadays, which could be activated by user's carelessness and give the remote hacker a chance to record user's activities, including his password.
2) Attain data through system flaws. The computer system storing and verifying password may have security flaws in different aspects, which would be used by hackers to hack into the system and obtain user's password files and decipher them by special tools.
3) Monitor through network data. As authentication information is usually transmitted through network, hackers could monitor these data by special tools and abstract user's name, password and other authentication information through analyzing.

The biggest problem of the traditional password authentication method is that the user's name, password and other key information used in authentication are transmitted on the internet with plain codes and stored in plain codes. In order to prevent problems mentioned above threatening the computer security, the common resolution is to use MD5 algorithm to convert the content of user's password information and transmit or store the results converted.

MD5 is the abbreviation of Message-Digest Algorithm 5, which was developed jointly by MIT Computer Science Laboratory and Ronald L. Rivest from RSA Data Security Inc. in early 90s in 20th century and evolved from MD2, MD3 and MD4. It compresses a piece of information with plain code and random length into 128 bits value by hash algorithm, which is called information distract. MD5 algorithm is irreversible and cannot recover the original plain code information from information abstraction, thus it is always believed safe.

However, some researches indicate that MD5 algorithm could be deciphered by collision attack, and the security of its application has received the challenge. This essay analyzes the application of MD5 algorithm in password authentication and its security, and probes into the physical measures of the application security of MD5 algorithm in password authentication.

ii) Hash Function And MD5 Algorithm

Hash function compresses a piece of information with random length by hash algorithm into fixed length value, which is called information abstraction. An information abstract generated from two pieces of different plain code is the so called collision. The requirements for safe Hash function include: first, two pieces of different plain code generate the same information abstract which should not be calculated and is called collision; second, a certain information abstract cannot be calculated through the other plain information generating the same abstract, which means the initial state cannot be deduced by the results. The so call "cannot be calculated" means that it is too expensive to get the results through the algorithm. Thus, Hush function usually contains two obvious features: first, no matter how long of the plain code information, the information abstract with certain length after calculating; second, if only the plain code information has any change, no matter how small the change is, the corresponding information abstract will be totally different. Thus, it is usually called "digital finger print", which could distinguish identities and ensure the unique and integrity of plain code information and the main effect is password authentication. MD5 algorithm is one of the most common Hash function. Its basic principle is to process the input information divided groups by 512 bits, and each group divided into 16 sub-groups with 32 bits. After a series of processing, the algorithm output composed by 4 groups with 32 bits, and cascade this 4 groups will generate a hash value with 128 bits. In the process of MD5 algorithm, fill information first to make its length 64 less than the multiple numbers of 512. The filling method is to attach a 1 and millions of 0, and add an information length before filling indicated by binary system with 64 bits. These two steps are to make the information length be the integer multiple of 512, and ensure the difference after different information filling.

iii) Application Security Analysis And Counter Measures Research

MD5 algorithm is mainly applied in digital signature, password authentication and other fields. For the sake of the password stored in database in the form of MD5 information abstract value, its security is threatened by MD5 being deciphered, and also by dictionary attack from hackers.

Considering that the attack to MD5 is mainly from collision attack, which means to find two different kinds of initial information, the values of calculated information abstract should be same. For example, presume a kind of initial information (M1) calculates the value of information abstract being H by MD5 algorithm, if the other kind of initial information (M2) is found, the value of information abstract by MD5 algorithm is still H, then M1 and M2 are a pair of collision. Deciphering MD5 is the process of looking for collision. Once MD5 is deciphered, the password stored in database in the form of MD5 information abstract value will be in danger.

Otherwise, there are 3 methods attacking the password stored in database in the form of MD5 information abstract value. First, look up MD5 information abstract value online. Some websites provide the look-up service for MD5 value online, after inputting MD5 value, if it is in the database, the password value could be obtained soon. Second, use MD5 deciphering tools. There are many special tools to decipher MD5 through dictionary setting. Third, obtain of reset the users' passwords through social engineering. Therefore, simple MD5 encryption cannot be absolute safe. To the attack mentioned above, MD5 algorithm processing could be exchanged, or use encryption algorithm, interference by add information to make the information abstract value processed by MD5 in database is not simple MD5 information abstract value anymore.

a) Exchange the MD5 Processing

- *Increase the Times of MD5 Processing*

The basic thought of this method is to conduct MD5 algorithm for N times to the user's password needing encryption, which means conduct MD5 algorithm again to the obtained information abstract value with 128 bits. The practical times should be decided by the administrator.

- *Intercept the Information Abstract Value*

The basic thought of the method is to use the user's password to conduct the MD5 algorithm first to get the MD5 information abstract value; and then select part of it (the first 18 bits for instance) to conduct the algorithm again to get the final result.

- *Divide Information Abstract Value*

The basic thought of the method is to use the user's password to conduct the MD5 algorithm first to get the MD5 information abstract value (presume the value is H); then divide H into two groups of 64 bits each on right and left sides, and conduct MD5 algorithm separately to get the corresponding information abstract value, shown as HL and HR; make HR and HL connect as an alphabetic string, and conduct MD5 algorithm again to get the final result.

iv) Additional Encryption Algorithm Interference

Before MD5 calculating, add an encryption algorithm to interfere the MD5 processing. The basic thought is to encrypt the user's password through self-defined encryption algorithm to get cipher code; get MD5 information abstract value by MD5 algorithm. There are countless kinds of self-defined encryption. Take user password "QDTONEWR" as an example to explain one of the kinds self defined encryption algorithm. Presume the key is "xsw".

Step1：Take the length of key as the index to make the plain code as a matrix (fill by spaces with insufficiency), the key and its length are self-defined. The results are like below:

Q D T
O N E
W R _
Step2：Conduct exclusive-or operation to the elements in each row and corresponding character;
x s w
↑↑↑
Q D T
O N E
W R _
The results are:
) 7 #
7 = 2
/ ! W
Step3： Form intermediate processing string by row priority
)7/7=!#2w;
Step4：Move left for n bits (n is the length of the key) to form cipher code,
7=!#2W)7/

v) Additional Information Interference

During MD5 processing, add an alphabetic string with certain content and interfere the processed data. Its basic thought is to make the user name input and system time as additional information, and conduct MD5 processing to "username + password + system time". To data through additional information interference processing, even the hacker decipher the MD5 algorithm, he is also difficult to lead out a comparison table with abundant characters from the dictionary established to decipher abundant users' passwords and decrease the possibility of password deciphered. Under the condition without revising the MD5 algorithm, the methods described above are the effective methods to increase the password security. If some flaws cause the user password data exposure, the MD5 exchange algorithm discussed above could increase the difficulty to decipher the password greatly and increase MD5 security in the password authentication application.

D. Implementation of Algorithm

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. The MD5 algorithm is an extension of the MD4 message-digest algorithm [12]. MD5 is slightly slower than MD4, but is more "conservative" in design. MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security. It incorporates some suggestions made by various reviewers, and contains additional optimizations. The MD5 algorithm is being placed in the public domain for review and possible adoption as a standard.

a) Terminology and Notation

In this document a "word" is a 32-bit quantity and a "byte" is an eight-bit quantity. A sequence of bits can be interpreted in a natural manner as a sequence of bytes, where each consecutive group of eight bits is interpreted as a byte with the high-order (most significant) bit of each byte listed first. Similarly, a sequence of bytes can be interpreted as a sequence of 32-bit words, where each consecutive group of four bytes is interpreted as a word with the low-order (least significant) byte given first.

Let x_i denote "x sub i". If the subscript is an expression, we surround it in braces, as in x_{i+1}. Similarly, we use ^ for superscripts (exponentiation), so that x^i denotes x to the i-th
power.

Let the symbol "+" denote addition of words (i.e., modulo-2^32 addition). Let X <<< s denote the 32-bit value obtained by circularly shifting (rotating) X left by s bit positions. Let not(X) denote the bit-wise complement of X, and let X v Y denote the bit-wise OR of X and Y. Let X xor Y denote the bit-wise XOR of X and Y, and let XY denote the bit-wise AND of X and Y.

b) MD5 Algorithm Description

We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

M_0 M_1 ... M_{B-1}

The following five steps are performed to compute the message digest of the message.

Step 1. Append Padding Bits11

The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.

Step 2. Append Length

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the low-order 64 bits of b are used.These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.
At this point the resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M[0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

Step 3. Initialize MD Buffer

A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 1 0

Step 4. Process Message in 16-Word Blocks

We first define four auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.
F(X,Y,Z) = XY v not(X) Z
G(X,Y,Z) = XZ v Y not(Z)
H(X,Y,Z) = X xor Y xor Z
I(X,Y,Z) = Y xor (X v not(Z))
In each bit position F acts as a conditional: if X then Y else Z.
The function F could have been defined using + instead of v since XY and not(X)Z will never have 1's in the same bit position.) It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of F(X,Y,Z) will be independent and unbiased.
The functions G, H, and I are similar to the function F, in that they act in "bitwise parallel" to produce their output from the bits of X, Y, and Z, in such a manner that if the corresponding bits of X, Y, and Z are independent and unbiased, then each bit of G(X,Y,Z), H(X,Y,Z), and I(X,Y,Z) will be independent and unbiased. Note that the function H is the bit-wise "xor" or "parity" function of its inputs.
This step uses a 64-element table T[1 ... 64] constructed from the sine function. Let T[i] denote the i-th element of the table, which is equal to the integer part of 4294967296 times abs(sin(i)), where I is in radians. The elements of the table are given in the appendix.
Do the following:
/* Process each 16-word block. */
For i = 0 to N/16-1 do
/* Copy block i into X. */
For j = 0 to 15 do
Set X[j] to M[i*16+j].
end /* of loop on j */
/* Save A as AA, B as BB, C as CC, and D as DD. */
AA = A

BB = B
CC = C
DD = D
/* Round 1. */
/* Let [abcd k s i] denote the operation
a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]
[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]
[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]
[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]
/* Round 2. */
/* Let [abcd k s i] denote the operation
a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]
[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]
[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]
[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]
/* Round 3. */
/* Let [abcd k s t] denote the operation
a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]
[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]
[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]
[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]
/* Round 4. */
/* Let [abcd k s t] denote the operation
a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */
/* Do the following 16 operations. */
[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]
[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]
[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]
[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]
/* Then perform the following additions. (That is increment each of the four registers by the value it had before this block was started.) */
A = A + AA
B = B + BB
C = C + CC
D = D + DD
end /* of loop on i */
Operations done for the 512 bit algorithm: there are new operations done on the each - 128 bit output to generate a independent 512 bit output. These operations are describes in followings figure 2:

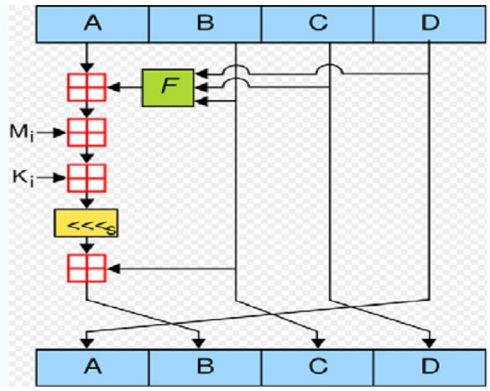

Figure 1: One MD5 Operation.

MD5 consists of 64 of these operations, grouped in four rounds of 16 operations[20]. F is a nonlinear functions used in each round. Mi denotes a 32-bit block of the message input, and Ki denotes a 32-bit constant, different for each operation. While <<<s denotes a left bit rotation by s places, s varies for each operation. Denotes addition mod 232 [21, 22].So, the code of 64 MD5's steps is shown as below:
FF(a,b,c,d, Mi,s, Ki)viz a=b+((a+(F(b,c,d)+ Mi + Ki)<<<s)
When 0<=i<=15
GG(a,b,c,d, Mi,s, Ki)viz a=b+((a+(G(b,c,d)+ Mi + Ki)<<<s)
When 16<=i<=31
HH(a,b,c,d, Mi,s, Ki)viz a=b+((a+(H(b,c,d)+ Mi + Ki)<<<s)
When 32<=i<=47
II(a,b,c,d, Mi,s, Ki) viz a=b+((a+(I(b,c,d)+ Mi + Ki)<<<s)
When 48<=i<=63
The following figure 3 describes that how the whole MD5 algorithm is implemented.

E. Implementation of a 512 Bit Algorithm by Using the Generic MD5 Algorithm

Its basic principle is to process the input information divided groups by 512 bits, and each group divided into 4 sub-groups with 128 bits. After a series of processing, the algorithm output composed by 4 groups with 128 bits, and cascade after certain operations this 4 groups will generate a hash value with 512 bits. In the process of MD5 algorithm, fill information first to make its length128 less than the multiple numbers of 512. The filling method is to attach a 1 and millions of 0, and add an information length before filling indicated by binary system with 64 bits. These two steps are to make the information length be the integer multiple of 512, and ensure the difference after different information filling.
Diagram to understand the new implemented algorithm:

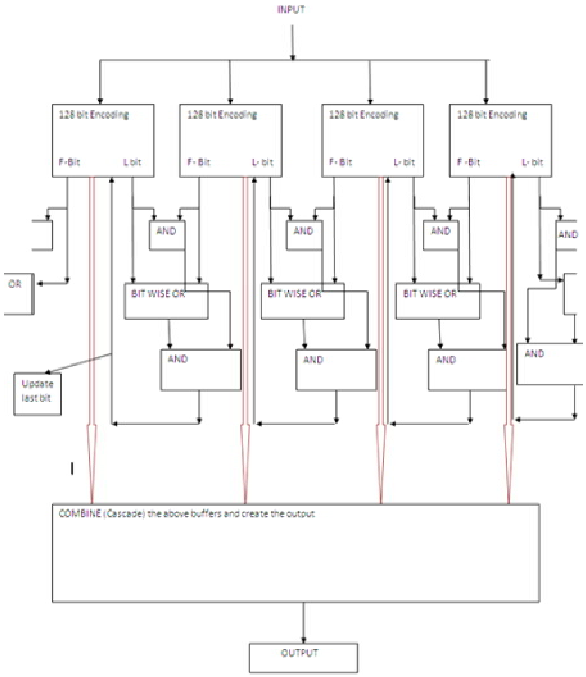

Figure 3: Diagram to understand the new implemented algorithm

Coding is done in C language. Code is available in Appendix A. For any input provided, it will generate the output of 512bits. The simulation results as follows:

INPUT:
41424344454647484 94A4B4C4D4E4F505152535455565758595A6162636465666768696A6B6C6D6E6F707172737475767778797A3031323334

OUTPUT:
"Áa\232\223~ àõ6\022\221>°\033\202Ü", '\0' <repeats 111 times>, "\026\036ElK\nÝzàþ3Ê¯ñ`Õ\b", '\0' <repeats 111 times>, "\0011µÈÙ³1\231;ÎÇ\a\022EG\205F", '\0' <repeats 111 times>, "\016\016Ô0O]{eúK\225\234â\034\r\035,", '\0' <repeats 111 times>, "\001"
INPUT:

"41424344454647484 94A4B4C4D4E4F505152535455565758595A6162636465666768696A6B6C6D6E6F7071727374
75767778797A3031323334C4D4E4F505152535455565758595A6162636444546474849 4A4B4C4D4E4F50515253545
554C4D4E4F50515253541323334C4D4E4F505152535455565758 59"

OUTPUT:
"B4¤Ñ|!\204\036¼~n¦é\035\017§", '\0' <repeats 111 times>, "136eïST³ÐÍA\017`ìoÐ\017", '\0' <repeats 111 times>, "\002º\016æ¶ù^\032}¬\a>´\025u\a", '\0' <repeats 111 times>, "0ð\2125×k\234£<w½\020\230Ç\203.Ú", '\0' <repeats 111 times>, "\002"

INPUT:
"41424344454647484 94A4B4C4D4E4F505152535455565758595A6162636465666768696A6B6C6D6E6F7071727374
75767778797A3031323334C4D4E4F505152535455565758595A6162636444546474849 4A4B4C4D4E4F50515253545
554C4D4E4F50515253541323334C4D4E4F505152535455565758594142434445464748494A4B4C4D4E4F505152535
455565758595A6162636465666768696A6B6C6D6E6F70717273747576778797A3031323334C4D4E4F50515253545
5565758595A6162636444546474849 4A4B4C4D4E4F50515253545554C4D4E4F505152535413233 34C4D4E4F50515
253545556575859"

OUTPUT:
"\237YÉf\024^JÐ®>Él\016¿ZA", '\0' <repeats 111 times>, "\001\001´\235ce×¹hàAÑó\033Ñl/", '\0' <repeats 112 times>, " z·Ü\215ß)w\206Õ\023\031¶\002V\036", '\0' <repeats 111 times>, "\001\215w·\032WäÑ\232ó;Ôm=\214¦G", '\0' <repeats 111 times>
There may be any input maximum of 512 bit will convert the encrypted output of 512 bit. The above mentioned three iterations are showing three different strings corresponding to three different inputs.
The output is encoded from the input. And the output would always is of 512bit message. In this way the secret information (e.g. passwords) can be shared with the peer. There is one more application of this algorithm is Message Authentication Code (MAC). This is an integrity check mechanism based on cryptographic hash functions using a secret key. Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties. In SCTP (Stream controlled transfer protocol), it is used by an endpoint to validate the State Cookie information that is returned from the peer in the COOKIE ECHO chunk.

II. CONCLUSIONS

This MD5 algorithm is for the 512 bit message transfer and also with the high security and Stream controlled transfer logic. This algorithm can be used in sending messages for 3G, 4G network. This can also be used for 5G network for which the work has been started. Here we are using 128 bit algorithm and using that as a basic element and create a application for 512 bit messages.
The output would always is of 512bit message. In this way the secret information (e.g. passwords) can be shared with the peer. There is one more application of this algorithm is Message Authentication Code (MAC). This is an integrity check mechanism based on cryptographic hash functions using a secret key. Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties. In SCTP (Stream controlled transfer protocol), it is used by an endpoint to validate the State Cookie information that is returned from the peer in the COOKIE ECHO chunk.
An Application of MD5 algorithm is implemented for the stream controlled transfer messages in the network. This would be a high security algorithm for data transfer in mobile networking with stream controlled logic. There may a vast number of applications for this algorithm in data transfer in various types of networks.

III.FUTURE WORK

In this thesis, we 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.
None of input messages that create same Hash Function in MD5-512bit algorithm cannot create same output in our algorithm (even by changing bits with OR, AND, Addition or Multiplication techniques).
We can extend our algorithm to have a bigger size of hash (768, 1024 …) like MD5 by extending the block size of compression functions or increasing number of them.

ACKNOWLEDGMENT

I am highly grateful to Dr. M. S. SAINI, Director, Guru Nanak Dev Engineering College, Ludhiana, for providing this opportunity to carry out the present work. I express my sincere gratitude to my Supervisor, Mr. Vivek Thapar, Assistant

Professor, Department of Computer Science & Engineering, GNDEC, Ludhiana who has been great help in carrying out this work, guiding and motivating with patience throughout the research work.
I would also like to express a deep sense of gratitude and thanks profusely to the department research committee members Mr. Akshay Girdhar, Mr. Parminder Singh, Mr. A. S. Brar and Mr. Amit Kamra, for providing me necessary guidance and encouragement during my paper work.

REFERENCES

[1] R. Rivest, "The MD5 Message-Digest Algorithm," RFC 1321, Apr. 1992.
[2] H. Dobbertin, A. Bosselaers and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD, Fast Software Encryption," LNCS 1039, pp. 71-92, Springer-Verlag, 1996.
[3] W. Stallings, Cryptography and Network Security, 2nd ed., Now York: Prentice-Hall, 1997.
[4] S. Dominikus, "A hardware implementation of MD4-family hash algorithms," Proc. 9th Int. Conf. on Electronics, Circuits and Systems, vol. 3, pp. 1143-1146, 2002.
[5] Chiu-Wah Ng, Tung-Sang Ng and Kun-Wah Yip "A UNIFIED ARCHITECTURE OF MD5 AND RIPEMD-160 HASH ALGORITHMS", Department of Electrical & Electronic Engineering, The University of Hong Kong Pokfulam Road, Hong Kong
[6] Anh Tuan Hoang, Katsuhiro Yamazaki and Shigeru Oyanagi "Multi-stage Pipelining MD5 Implementations on FPGA with Data Forwarding", Ritsumeikan University, 1-1-1 Noji Higashi, Kusatsu, Shiga 525-8577, Japan
[7] Changxin Li1, Hongwei Wu2, Shifeng Chen1, "Efficient Implementation for MD5-RC4 Encryption", Xiaochao Li2* and Donghui Guo1,2
[8] Guang Hu, Jianhua Ma and Benxiong Huang, "High Throughput Implementation of MD5 Algorithm on GPU" Guang Hu, Department of Electron and Information, Huazhong University of Science and Technology, Wuhan, China huguang@mail.hust.edu.cn, Jianhua Ma, Faculty of Computer and Information Sciences, Hosei University, Tokyo 184-8584, Japan, jianhua@hosei.ac.jp
[9] Benxiong Huang, Department of Electron and Information, Huazhong University of Science and Technology, Wuhan, China, huangbx@mail.hust.edu.cn
[10] Alok Kumar Kasgar, Jitendra Agrawal, Santosh Sahu , "New Modified 256-bit MD5 Algorithm with SHA Compression Function ", School of IT School of IT School of IT Rajiv Gandhi Technical University Rajiv Gandhi Technical University Rajiv Gandhi Technical University Bhopal (M.P.) Bhopal (M.P.) Bhopal (M.P.)
[11] Chiu-Wah Ng, Tung-Sang Ng and Kun-Wah Yip "A UNIFIED ARCHITECTURE OF MD5 AND RIPEMD-160 HASH ALGORITHMS", Department of Electrical & Electronic Engineering, The University of Hong Kong Pokfulam Road, Hong Kong.
[12] 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.
[13] 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.
[14] NIST FIPS PUB 180-1. Oct. 2001.
[15] NIST, "Secure Hash Standard (SHS)", FIPS PUB 180-2, 2002.
[16] S. Chang, M. Dworkin, Workshop Report, The First Cryptographic Hash Workshop, Report prepared, NIST 2005.
[17] E. Biham, R. Chen, "New results on SHA-0 and SHA-1" Crypto 2004 Rump Session, Aug. 2004.
[18] K. Matusiewicz and J. Pieprzyk "Finding good differential patterns for attacks on SHA-1" eprint 2004 Available : http://eprint. iacr. org/2004/364. pdf.
[19] 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.
[20] Rivest R L. The MD5 message digest algorithm [EB/OL].
[21] Xiaoyun Wang, Dengguo, k. , m. , m, HAVAL-128 and RIPEMD], Cryptology ePrint Archive Report 2004/199, 16 August 2004,
[22] J. Black, M. Cochran, T. Highland: A Study of the MD5 Attacks: Insights and Improvements, March 3, 2006
[23] Tao Xie and DengguoFeng (30 May 2009). How to Find Weak Input Differences for MD5 Collision Attacks.
[24] ChristofPaar, Jan Pelzl, Bart Preneel (2010). Understanding Cryptography: A Textbook for Students and ractitioners. Springer. p. 7. ISBN 3642041000.
[25] M. E. Hellman, H. R. Amirazizi, "A Cryptanalytic Time - Memory Trade-Off," IEEE Transactions on Information Theory, vol. 34-3, pp. 505-512, 1988
[26] 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/

