

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

Implementation of Md5- 640 Bits Algorithm

Deepika Sharma¹

Department of Computer Science & Engineering
Somany (P.G.) Institute of Technology & Management
Rewari – India

Pushpender Sarao²

Department of Computer Science & Engineering
Somany (P.G.) Institute of Technology & Management
Rewari – India

Sunita Dudi³

Department of Computer Application
M.D. Goenka Girls College
Laxmangarh (Sikar) – India

Abstract: *In network security & cryptography Rivest has given various message digest algorithms like MD2, MD4, and MD5. These algorithms are basically designed for high security purpose where a large message has to be "compressed" in a secure manner before being signed with the private key. All above algorithms take a message of arbitrary length and produce a 128-bit message digest. MD5 has been used in a wide variety of security applications, and mainly used to check data integrity. In proposed algorithm going to implement the MD5 algorithm for the 640 bit message transfer with the high security encryption standards. 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 as a basic element and create an application for 640 bit messages.*

Keywords: *Cryptography; Hash Function; MD5; Network simulator; Message digest.*

I. INTRODUCTION

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. 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. MD2, MD4, and MD5 are message-digest algorithms developed by Rivest. They are meant for digital signature applications where a large message has to be "compressed" in a secure manner before being signed with the private key. All three algorithms take a message of arbitrary length and produce a 128-bit message digest. However it has been proved that 128 bit hash output does not offer sufficient security protection in current high speed and advanced network. The size of the hash 128 bit is small enough to contemplate a birthday attack.

MD5 is widely used in several public key cryptographic algorithms and Internet communication in general so to provide higher security protection, an application of MD5 algorithm is implemented in the network which produces 640-bit message digest. This would be a high security algorithm for data transfer in mobile networking. There may be vast number of

applications for this algorithm in data transfer in various types of networks. In the proposed newly implemented algorithm, MD5 – 640 bits is developed. First the input message is padded and divides into data blocks of length 640 bits. Each data block is treated as 20, 32 bit words. The algorithm iteratively processes each data block. For the first data block, an initial value is used to compute an intermediate results which is called chaining variable. That is updated on the basis of previous results and input data block. After all iterations are done, the final chaining variable is known as hash value.

II. MESSAGE DIGEST5-125 BIT ALGORITHM

MD5 algorithm was given by Ronald L.Rivest. Its predecessor is MD, MD2, MD3 and MD4. It can compress any length of data into an information digest of 128 bits. This algorithm makes use of a series of non- linear algorithm to do the circular operation, so that crackers cannot restore the original data. MD5 is an irreversible transformation transforming a set of data of any length into a hash value of 128 – bit length. This algorithm is based on message length.

Step 1:- Padding bits and Append Length

Padding of the bits is compulsory with '0' and '1' first and last respectively until the resulting \neq bit length which = $448 \bmod 512$, and the last of bit length of the original message as 64-bit integer. The last bit length of the message which is already padded is $512N$ for a true integer N .

Step 2:- Divide the input into 512-bit blocks

The message which is already padded is now partitioned into N successive 512-bit blocks m_1, m_2, \dots, m_n .

Step 3:- Initialize Channing variables Initialization of 32-bit number in the form of chaining

Variables (A,B,C,D) these values are represented in hash only

A = 01 17 2d 43

B = 89 AB CD EF

C = FE DC BA 98

D = 76 54 32 10

Step 4:- Process blocks

The heart of MD5 is an algorithm which is used for the processing of the message. The message M is divided into 512-bit blocks which are processed separately. Let X_j denote the j th block of M . First, X_0 , i.e. the lowest 512-bits of M , is processed with the algorithm, then X_1 etc., until the entire M is processed. The algorithm consists of four rounds, each of which comprise 16 steps. Hence, 64 steps are performed in the algorithm. Let i be the index of a step. Let $X_j[k]$ denote the k^{th} 32-bit word of X_j and let $T[i]$ be a table of 64 32-bit constants. Let $\ll s$ denote circular shift left by s bits. Values of k , s and $T[i]$ depend on i . The algorithm is performed as follows:

First, values of A , B , C and D are stored into temporary variables AA , BB , CC and DD . Then, the following operations are performed –

for $i = 0$ to 63:

$$A = B + ((A + \text{Func}(B, C, D) + X_j[k] + T[i]) \ll s)$$

$$A \leftarrow D, B \leftarrow A, C \leftarrow B, D \leftarrow C$$

Where additions are additions of words, i.e. additions modulo-232. $\text{Func}(X, Y, Z)$ is different for every round. Function $F(X, Y, Z)$ is used for the first round ($0 \leq i \leq 15$), $G(X, Y, Z)$ for the second ($16 \leq i \leq 31$), $H(X, Y, Z)$ for the third ($32 \leq i \leq 47$) and $I(X, Y, Z)$ for the final round ($48 \leq i \leq 63$). The functions are defined as follows:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Where \vee is a bitwise or-operation, \neg is a bitwise complement, \oplus is a bitwise exclusive-or-operation (xor) and \wedge is a bitwise and-operation.

$$A = A+AA$$

$$B = B+BB$$

$$C = C+CC$$

$$D = D+DD$$

Step 5:- Hashed Output

Then all X_j have been processed with the algorithm, the message digest of M is in A , B , C , and D . The low-order byte of A is the first byte of the digest and the high-order byte of D is its last byte. There are 4 rounds performed in message digest 5 (MD5) which is of 128 bits.

III. TECHNOLOGY USED

Socket programming present in a Linux OS is used for the implementation of the proposed work. Here the C language is going to be used in the implementation of algorithm. The simple C compiler on Linux OS is required.

A. Socket Programming

A Socket is an end point of communication between two systems on a network. In other words socket is a combination of IP address and port on one system. So on each system a socket exists for a process interacting with the socket on other system over the network. A combination of local socket and the socket at the remote system is also known a 'Four tuple' or '4-tuple'. Each connection between two processes running at different systems can be uniquely identified through their 4-tuple.socket is the place where a computer gathers and puts data into the internet. To create a socket a program has to specify socket type and address domain. If two processes have same type of socket and domain then only they can communicate.

B. The Client-Server model

Most *client server model* uses interprocess communication. The term refers to the two terminal processes which communicate with each other via network. One of the two processes is, the *client*, establish the connection with other one, the *server*, which responds to the request for information. A good example is a person who makes a phone call to another person. In such model it is necessary for client to know the existence and the logical address of the server, but the server does not need to know the address even if the existence of the client prior to the connection being established. After a connection is established, both sides can exchange the information with each other. The system calls for establishing a connection is little bit different for the client and the server, but both involve the basic construct of a *socket*.

C. C Programming

C is a programming language which was given by Dennis Ritchi. To develop New modify MD5- 640 bit algorithm C language is used to do socket programming and to implement the logic for 640 bit message digest.

D. C Compiler

For compiling and executing C code to create client, server and socket among them, a C compiler is strictly required. On Linux platform there are number of c compilers available, out of them gcc compiler is used. Using Gcc compiler code of client and server can be run on terminal which performs whole logic of the algorithm via creating the socket. The GNU Compiler Collection (GCC) is a compiler system. It was created by the GNU Project supporting various programming languages such as C (gcc), C++ (g++) etc.

E. Linux operating system

It is an operating system which is more secure and reliable in compare of other operating systems. The most obvious advantage of using linux is free or very less costly as well as more flexible in compare of other operating systems. Second main advantage is that Linux system rarely crash, and when they do, the whole system does not go down. so due to above main reasons the proposed algorithm is developed on linux platform which is most secure operating system.

Ubuntu 14.04 LTS.

Ubuntu is a linux operating system which is more secure than any other operating system. Here the abbreviation LTS stand for "Long term support". Ubuntu is a fantastic Linux choice because it has the largest user base, meaning as a user won't be left behind by developers. Additionally, Ubuntu is a Linux distribution that is the base of choice for a lot of other popular distributions. The combination of these two reasons means that any software that's made for Linux will almost always be available for Ubuntu, especially games.

IV. NEWLY IMPLEMENTED MD5-64 BIT ALGORITHM

Its basic principle is to process the given input for encryption by dividing the group of 640 bits message digests as a output, and each group is divided into 5 subgroups with 128 bits. After applying a series of operations on whole groups, the algorithm generates a hash value output of 640 bits. Figure 1 is the block diagram of newly implemented algorithm.

A. Algorithm Description-

Initially suppose that we have a z- bit message as input and going to apply this algorithm to generate 640 bit message digest. The following five steps are performed to for applying encryption-

Step 1 – Extending bits

The message is extended or padded so that its length is congruent to 576 modulo 640.that is ,the message is extended so that its only 64 bits generates a multiple of 640 bits long. In all , at least one bit and at most 640 bits are appended.

Step 2- Extending the length of data

A 64 bit representation of z (original length) is extended to the result of the previous step. These bits are appended as two 32-bit words and appended low order word first in accordance with he previous conventions. At this point the output has a length exact multiple of 640 bits.

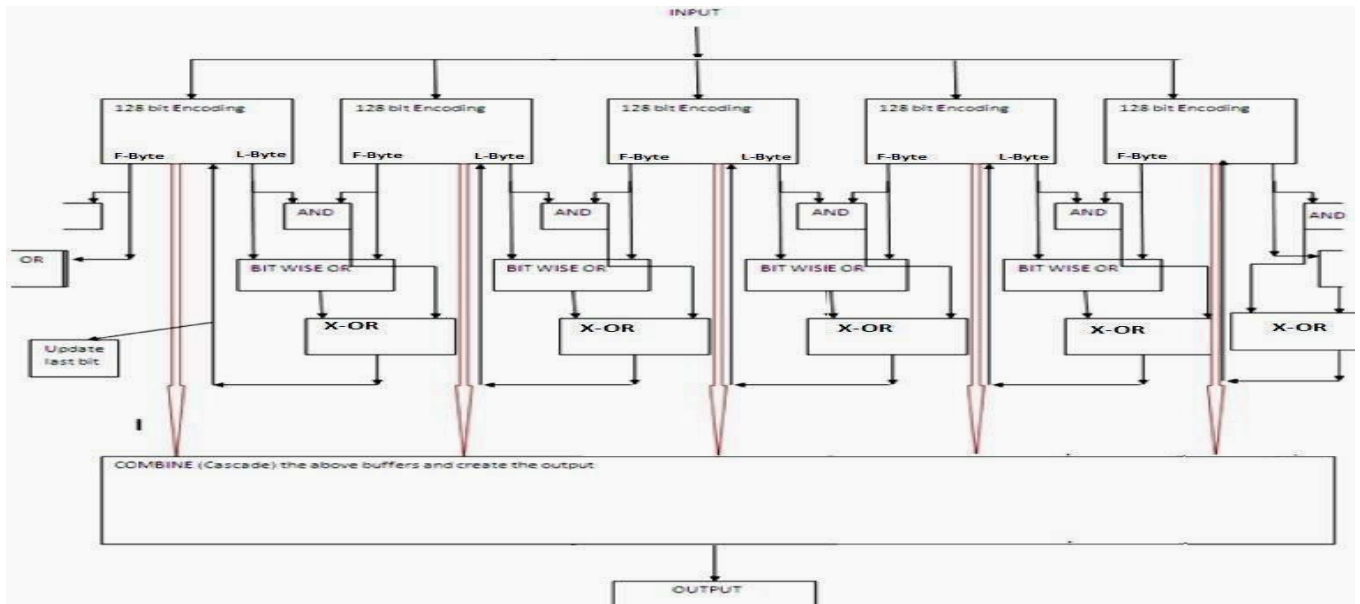


Fig. 1 Diagram for newly implemented algorithm

Step 3-Initialize MD5 standard parameters and Bit operation function

First divide the input into five group of 128 bits. Now for performing operations take Last byte of first group's output and first byte of second group's output, perform AND operation on it. Also perform bitwise OR operation on it. Then apply X-OR operation on output of both operations.

```
lc1 = *(yloc_oup_0 + ((KEY_SIZE/8)-1));
```

```
lc2 = *(yloc_oup_1);
```

```
A_cr_op_1 = (lc1 & lc2);
```

```
O_cr_op_1 = (lc1 | lc2);
```

```
cr_op_1_0 = (A_cr_op_1 & O_cr_op_1);
```

```
*(yloc_oup_0 + (KEY_SIZE/8) - 1) = cr_op_1_0;
```

In similar fashion do same operation on all groups. At last perform bitwise addition on output of all results. It is called bitwise streaming of the result. Following this logic implements MD5 – 640 bit algorithm. This is highly secure and advance in compare of MD5- 128 bit algorithm.

Step 4- Main Encryption Process

At client Terminal-

Please Enter the input WITHOUT CHARACTERS (Not greater than 640 bits OR 64 bytes (characters))

deepikasharma

Please Enter the input (again): deepikasharma

There are 1 iterations for every input(after modification of few characters)

Input Entered as deepikasharma

Output Obtained is:

0 byte = e3, 1 byte = 63, 2 byte = d0, 3 byte = 8f, 4 byte = 77, 5 byte = ea, 6 byte = cc, 7 byte = 4f, 8 byte = 45, 9 byte = 1e, 10 byte = 50, 11 byte = 4c, 12 byte = 16, 13 byte = ac, 14 byte = 2c, 15 byte = 10, 16 byte = 76, 17 byte = 4d, 18 byte = 4b, 19 byte

```
= fb, 20 byte = 34, 21 byte = cd, 22 byte = 4b, 23 byte = e6, 24 byte = b5, 25 byte = 60, 26 byte = 68, 27 byte = 68, 28 byte = 5b, 29 byte = 26, 30 byte = 31, 31 byte = 15, 32 byte = 9d, 33 byte = bb, 34 byte = 27, 35 byte = a7, 36 byte = 3f, 37 byte = 15, 38 byte = 94, 39 byte = 15, 40 byte = 7e, 41 byte = c1, 42 byte = 14, 43 byte = f0, 44 byte = 6a, 45 byte = 69, 46 byte = c, 47 byte = 4, 48 byte = c, 49 byte = cc, 50 byte = 37, 51 byte = e3, 52 byte = 31, 53 byte = 1, 54 byte = 3e, 55 byte = 61, 56 byte = 60, 57 byte = bc, 58 byte = a5, 59 byte = f0, 60 byte = cd, 61 byte = 70, 62 byte = d4, 63 byte = 2, 64 byte = 52, 65 byte = b, 66 byte = 2c, 67 byte = 86, 68 byte = 9a, 69 byte = fc, 70 byte = 67, 71 byte = 12, 72 byte = f8, 73 byte = 3a, 74 byte = f2, 75 byte = a6, 76 byte = 37, 77 byte = 1c, 78 byte = 16, 79 byte = c0,
```

Congratulations Encrypting SUCCESSFULL

EXITING APPLICATION

At server Terminal-

```
deepika@deepika-VPCEH26EN:~$ cd /home/deepika/Desktop/crypto/server
```

```
deepika@deepika-VPCEH26EN:~/Desktop/crypto/server$ ./server
```

Listening

Data received

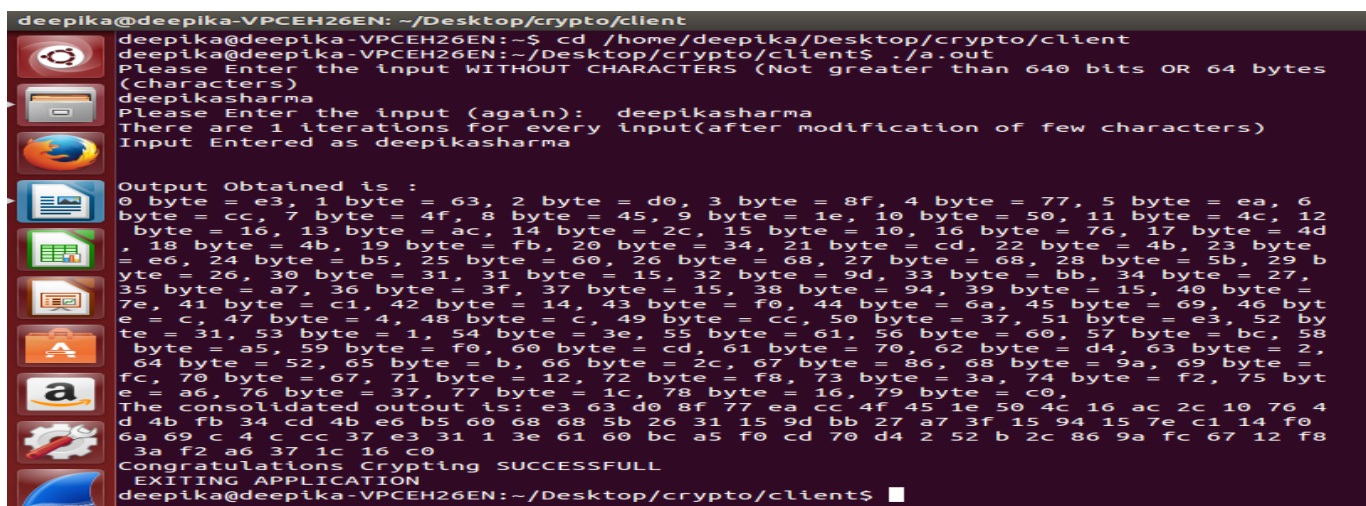
```
e3 63 d0 8f 77 ea cc 4f 45 1e 50 4c 16 ac 2c 10 76 4d 4b fb 34 cd 4b e6 b5 60 68 68 5b 26 31 15 9d bb 27 a7 3f 15 94 15 7e c1 14 f0 6a 69 c 4 c cc 37 e3 31 1 3e 61 60 bc a5 f0 cd 70 d4 2 52 b 2c 86 9a fc 67 12 f8 3a f2 a6 37 1c 16 c0
```

V. RESULTS

Implementation of New MD5- 640 Bits Algorithm is done in C language using socket programming on terminal. Platform used is Ubuntu 14.04 LTS. For compilation and establishment of socket between client & server gcc compiler is used in Linux. When client & server starts communication with each other in reality, message is transferred from client to server in form of data packets. These data packets deliver in encrypted form. The logic behind encryption of data packets is based on basic of MD5 algorithm but with newly implemented MD5- 640 bits algorithm.

A. At Client Terminal-

First of all c coding is done to implement the message digest5 algorithm for 640 bits. When all c programming to implement the algorithm is done then client code is run to create socket from client side. The following window is terminal screen of client side. In this terminal folder, client code is compiled. On doing so a binary file is generated in client folder. Then we execute this binary file by writing ./a.out . On successful execution of client socket algorithm starts its execution by asking for the input, and converting the input in to encrypted 640 bit output.



```
deepika@deepika-VPCEH26EN: ~/Desktop/crypto/client
deepika@deepika-VPCEH26EN:~$ cd /home/deepika/Desktop/crypto/client
deepika@deepika-VPCEH26EN:~/Desktop/crypto/client$ ./a.out
Please Enter the input WITHOUT CHARACTERS (Not greater than 640 bits OR 64 bytes
(Character))
deepikasharma
Please Enter the input (again): deepikasharma
There are 1 iterations for every input(after modification of few characters)
Input Entered as deepikasharma

Output Obtained is :
0 byte = e3, 1 byte = 63, 2 byte = d0, 3 byte = 8f, 4 byte = 77, 5 byte = ea, 6
byte = cc, 7 byte = 4f, 8 byte = 45, 9 byte = 1e, 10 byte = 50, 11 byte = 4c, 12
byte = 16, 13 byte = ac, 14 byte = 2c, 15 byte = 10, 16 byte = 76, 17 byte = 4d
, 18 byte = 4b, 19 byte = fb, 20 byte = 34, 21 byte = cd, 22 byte = 4b, 23 byte
= e6, 24 byte = b5, 25 byte = 60, 26 byte = 68, 27 byte = 68, 28 byte = 5b, 29 b
yte = 26, 30 byte = 31, 31 byte = 15, 32 byte = 9d, 33 byte = bb, 34 byte = 27,
35 byte = a7, 36 byte = 3f, 37 byte = 15, 38 byte = 94, 39 byte = 15, 40 byte =
7e, 41 byte = c1, 42 byte = 14, 43 byte = f0, 44 byte = 6a, 45 byte = 69, 46 byt
e = c, 47 byte = 4, 48 byte = c, 49 byte = cc, 50 byte = 37, 51 byte = e3, 52 by
te = 31, 53 byte = 1, 54 byte = 3e, 55 byte = 61, 56 byte = 60, 57 byte = bc, 58
byte = a5, 59 byte = f0, 60 byte = cd, 61 byte = 70, 62 byte = d4, 63 byte = 2,
64 byte = 52, 65 byte = b, 66 byte = 2c, 67 byte = 86, 68 byte = 9a, 69 byte =
fc, 70 byte = 67, 71 byte = 12, 72 byte = f8, 73 byte = 3a, 74 byte = f2, 75 byt
e = a6, 76 byte = 37, 77 byte = 1c, 78 byte = 16, 79 byte = c0,
The consolidated outout is: e3 63 d0 8f 77 ea cc 4f 45 1e 50 4c 16 ac 2c 10 76 4
d 4b fb 34 cd 4b e6 b5 60 68 68 5b 26 31 15 9d bb 27 a7 3f 15 94 15 7e c1 14 f0
6a 69 c 4 c cc 37 e3 31 1 3e 61 60 bc a5 f0 cd 70 d4 2 52 b 2c 86 9a fc 67 12 f8
3a f2 a6 37 1c 16 c0
Congratulations Encrypting SUCCESSFULL
EXITING APPLICATION
deepika@deepika-VPCEH26EN:~/Desktop/crypto/client$ █
```


B. At server terminal-

Above is the terminal screen for server side. When the client needs to communicate with server then before data transmission it needs to establish connection with server. So to create socket from server side first compile the code of server folder then a binary file generates in that folder. Now execute this one. It activates the server and it starts listening the network.

```

deepika@deepika-VPCEH26EN: ~/Desktop/crypto/server
deepika@deepika-VPCEH26EN:~$ cd /home/deepika/Desktop/crypto/server
deepika@deepika-VPCEH26EN:~/Desktop/crypto/server$ ./server
Listening
Data received
e3 63 d0 8f 77 ea cc 4f 45 1e 50 4c 16 ac 2c 10 76 4d 4b fb 34 cd 4b e6 b5 60 68
68 5b 26 31 15 9d bb 27 a7 3f 15 94 15 7e c1 14 f0 6a 69 c 4 c cc 37 e3 31 1 3e
61 60 bc a5 f0 cd 70 d4 2 52 b 2c 86 9a fc 67 12 f8 3a f2 a6 37 1c 16 c0
deepika@deepika-VPCEH26EN:~/Desktop/crypto/server$

```

C. Communication between client & server –

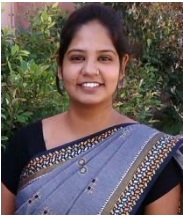
Now connection is established between client and server now client can give any data as an input, of any length for transmission. The logic works which provides output of 640 bits message digest and data packets reaches at server side in encrypted form as shown in client terminal and server terminal window.

VI. CONCLUSION AND FUTURE WORK

It is a newly implemented algorithm for 640 bit message digest. It provides high security in data transfer. This algorithm can be used for 3G, 4G even if for 5G also for which the work has been started. For the implementation of this algorithm we are using message digest5 - 128 bit algorithm as a basic element and create an application for 640 bit messages. The output would always be of 640 bit message. In this way the secret information like passwords can be shared with the peer. One more application of this algorithm is message authentication code (MAC). The message authentication codes are used between two terminals who share a secret key to govern secure data transmission between them. By using this whole concept an application of MD5 algorithm is implemented which is highly secured for the data packets transmission in the network. In future we can extend this 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.

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., New 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. "A UNIFIED ARCHITECTURE OF MD5 AND RIPEMD-160 HASH ALGORITHMS" Chiu-Wah Ng, Tung-Sang Ng and Kun-Wah Yip Department of Electrical & Electronic Engineering, The University of Hong Kong Pokfulam Road, Hong Kong
6. "Multi-stage Pipelining MD5 Implementations on FPGA with Data Forwarding", Anh Tuan Hoang, Katsuhiro Yamazaki and Shigeru Oyanagi, Ritsumeikan University, 1-1-1 Noji Higashi, Kusatsu, Shiga 525-8577, Japan
7. "Efficient Implementation for MD5-RC4ncryption", Changxin Li1, Hongwei Wu2, Shifeng Chen1, Xiaochao Li2* and Donghui Guo1,2 1. Dept. of Physics, Xiamen University, Fujian 361005, China 2. Dept. of Electronic Engineering, Xiamen University, Fujian 361005, China
8. "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 Benxiong Huang, Department of Electron and Information, Huazhong University of Science and Technology, Wuhan, China, huangbx@mail.hust.edu.cn
9. "New Modified 256-bit MD5 Algorithm with SHA Compression Function" Alok kumar kasgar, Jitendra Agrawal, Santosh Sahu, 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.)

AUTHOR(S) PROFILE

Deepika Sharma, is M.Tech Scholar of Somany (P.G.) Institute of Technology & Management, Rewari (HR), India. She has been awarded by degree of Master of Computer Application By Mody Institute of Technology & Science in 2012. She has contributed her work in the area of research and development in Engineering. Presented and published several research papers in International and National Conferences and Journals.



Pushpender Sarao, is working as Dean Academics and Assistant Professor in Somany (P.G.) Institute of Technology & Management, Rewari (HR), India. He is Pursuing Ph.D. (CSE) from Shri Venkateshwara University, Gajraula, Amroha (UP) and has been awarded by degree of Master of Technology (CSE) by M.D.U. Rohtak (HR). He has published several research papers in International and National Conferences and journals.



Sunita Dudi, is working as Lecturer in M.D. Goenka Girls College, Laxmangarh (Sikar) Rajasthan, India. She has been awarded by degree of Master of Computer Application by Rajasthan University, Jaipur. She is a good academician and contributed many work in the area of research and development in Computer Science.