An Extended MD5 (ExMD5) Hashing Algorithm For Better Data Integrity

S. Karthikeyan

Abstract

Recent developments in scientific and engineering applications communication are playing a major role in online transactions. All the transactions across the globe are shared by network. The integrity ensures that only authorized parties are able to modify the transmitted information. Modification includes writing, changing, changing status and delaying or replaying transmitted messages. The Message Digest functions MD4 and MD5 provide one-way function integrity in the form of octets, 32-bit and 64 bit words respectively. The extended MD5 (ExMD5) was designed to be somewhat more conservative than MD5 in terms of being more concerned with security. An extended MD5 hashing algorithm that follows five passes than MD5, which follows only four passes. In the proposed approach, the message is processed in 512-bit blocks and the message digest is a 128-bit quantity. Each stage consists of computing function based on the 512-bit message chunk and the message digest to produce a new intermediate value. The value of the message digest is the result of the output of the final block of the message. An extended MD5 hashing provides the better data integrity and the resulted values are compared against existing hashing algorithms such as MD4 and MD5 algorithms.

Key Words : Hashing Algorithon, one-way function, message digest and integrity.

Reader and Head, Department of Computer Science, Karpagam Arts and Science College (Autonomous), Coimbatore-21.

1. Introduction

The Trusted Network Interpretation [4] points out that integrity ensures computerized data, which are the same as those in source documents and have not been exposed to accidental or malicious alteration or destruction. The integrity ensures the data related are precise, accurate, unmodified, meaningful and usable. The operating system, database management system and the network enforce the integrity.

The integrity of data is more important like confidentiality. In some situations passing authentication data, the integrity is paramount. In other cases, the need for integrity is less obvious. Some of the message integrity threats are falsification of messages and noise. The hackers may do the falsification of messages during transmission using active wiretap. Signals sent over communication media are subject to interference from other traffic on the same media, as well as from natural sources such as, lightning, and electric motors. Such unintentional interference is called noise. These forms of noise are inevitable, and it can threaten the integrity of data in a message.

The integrity defines one-way hash function generating the checksum of the message [2]. When the receiver gets the data, it hashes it as well and compares the two sums, if they match, then the data is unaltered. The integrity is implemented through the use of Message Authentication Code (MAC) and hash or Message Digest Functions. The MAC is cryptographically generated fixed length quantity and associated with a message to reassure the recipient that the message is genuine. The Message Digest functions MD2, MD4 and MD5 provide one-way function in the form of octets, 32-bit and 64 bit words respectively. The examples for message digest functions are MD2, MD4 and MD5. The MD5 was designed to be somewhat more conservative than MD4 in terms of being more concerned with security.

1.1. Hash Function

A hash function maps a message of any length into a fixed length hash value or message digest. The hash value depends on input value. It provides an error detection capability [3]. A cryptographic function, such as DES or AES, is especially appropriate for sealing values, since the outsider will not know the key and thus will not be able to modify the stored value. A change in one bit or bits in the message results in a change in the hash code. A hash function is an easily computable map $f \rightarrow x\ h$ from a very long input 'x' to a much shorter output 'h'. In a hash function it is not computationally feasible to find same hash value for two different inputs x and x^1 such that $f(x) = f(x^i)$. The most widely used cryptographic hash functions are MD4, MD5 and Secure Hash Algorithm (SHA). The authors [5] will describe the following properties that are important for hash function 'h'.

- The function 'h' can be applied to a block of data of any size.
- The function 'h' produces a fixed-length output.
- The h(x) is relatively easy to compute for any given 'x'.
- For any given value 'h', it is computationally infeasible to find 'x' such that h(x) = h.
- For any given block 'x', it is computationally infeasible to find y '" x such that h(y) = h(x).

The main usage of hash function 'h' is to maintain confidentiality and authentication between sender and receiver. The simplest hash function is the bit-by-bit exclusive-OR (XOR) of every block. This operation can be expressed as follows:

$$M_i = C_{i1} \oplus C_{i2} \ldots C_{in}$$

Where,

M_i = ith bit of the hash code, $1 \leq i \leq n$

n = Number of n-bit blocks in the input

C_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity for each bit position. This process is known as longitudinal redundancy check.

2. Literature Review

The following various hash functions and its usages are discussed by the authors [9].

The hash function $S \longrightarrow R : M \| E(PR_s, H(M))$ provides authentication and digital signature [7]. The hash code of the message 'M' is encrypted using private key of the sender PR_s. Thus, the $E(PR_s,H(M))$ is an encryption function of a variable length message 'M' and the private key PR_s and it produces a fixed size output. The receiver decrypts hash value, which is received from sender using public key of the sender PU_s and compares with the original message M. If it is equal the integrity is proved.

The hash function $S \longrightarrow R : E(K,(M \| H(M \| SV)))$ provides authentication and confidentiality. The technique assumes that the two communicating parties share a common secret value (SV) [11]. The sender 'S' computes the hash values over the concatenation of 'M' and 'SV' and appends the result hashing value to 'M'. Then entire message with hash code is encrypted using shared key value 'K'. The receiver decrypts entire encoded message using shared secret key value 'K'. The receiver extracts the original message 'M' and concatenates with secret value. Calculate the hash value for 'M||SV' and compare with received hash value. In this approach both authentication and confidentiality is well proved. The various authors discuss the different approaches related to the hashing algorithm.

Anderson and Math [9] provide the information on classifications of hash functions. There are many applications for which one-way hash functions are required. Digital signatures are one example; it is usually not practical to sign the whole message, as public key algorithms are rather slow, so the normal practice is to hash a message to a digit between 128 ad 160 bits, and sign this need. The paper describes the real requirements of hash functions like collision free, complementation free, addition free and multiplication freedom properties. These freedom properties are central to controlling interactions between cryptographic algorithms, and have the potential to be useful in algorithm design as well.

Anderson and Biham [8] present a new hash function, which is called a Tiger, to be designed for 64 bit processors and secure than MD4, SHA-1 and Snefru-8. The next generation of processors has 64-bit words and the older hash functions could not be implemented efficiently. The Tiger hash function is stronger and faster than SHA-1 in 32-bit processors and about three times faster on 64-bit processors. It outputs 192-bit hash value, which can be truncated to 128-bit or 168-bits for existing applications compatibility. However this Tiger hash function has more passes than existing hash functions.

The Bellare, Canetti and Krawczyk [6] present new, simple and practical constructions of message authentication schemes based on a cryptographic hash function. This paper describes basic properties of cryptographic hash functions and keyed hash functions. It also describes various attacks in message authentication codes and provides the solution.

The Bosselaers [1] presents a performance improvement 15% for the MD4 family of hash functions. The improvement is obtained by substituting n-cycle instructions by n-1 cycle instructions and reducing so many instructions. This paper also compares the performance improvement on MD4, MD5, and SHA-1. This comparison is done by 90 MHz Pentium processors.

Challal, Bouabdallah and Bettahar [12] describe the hybrid hash-chaining scheme in conjunction with an adaptive and efficient data source authentication protocol, which tolerates packet loss and guarantees non-repudiation of media-streaming origin. The hybrid hash chaining has the following terminology: If a packet P_j contains the hash of a packet P_i, the hash link connects P_i to P_j. The target of P_i is P_j. A signature packet is a sequence of packet hashes, which are signed using a conventional digital signature scheme. A hash link relates the packet with signature packets. This protocol allows saving the bandwidth and improves the probability that a packet be verifiable even if some packets are lost.

Cao, Lin and Xue [10] provide the information on secure randomized RSA-based blind signature scheme. The blind signature scheme can yield a signature and message pair whose information does not leak to the signer. When blind signatures are used to design e-cash schemes there are two problems: double spending and accuracy of signer information. The proposed scheme in this paper satisfies blindness and unforgeability properties. The computation cost of this scheme is six modular exponentiations, six modular multiplications, three hashing operations and twice of random number generation performed by the user to obtain and verify a signature. This paper uses one-way public hash function like MD4, SHA-1.

3. Extended Md5 Hashing Algorithm

The Extended MD5 algorithm (ExMD5) takes as input a message of random length and produces as output a 128-bit message digest of the input. This algorithm ensures that it is computationally infeasible to produce two messages having the same message digests, or to produce any message having a given pre-specified target message digest. The extended MD5 algorithm is an extension of the MD5 message digest algorithm. The proposed algorithm is more secure and conservative in design than MD5. It is intended for digital signature applications, where it identifies the correct sender.

The extended MD5 algorithm is designed to be quite secure comparing to existing hash algorithms like MD2, MD4, and MD5. In addition, it does not require any large substitution tables; the algorithm can be coded quite compactly. The extended MD5 algorithm is similar to MD4 and MD5. The major differences are:

- ✦ MD4 makes three passes over each 16-octet chunk of the message and MD5 has four passes for every 16-octet chunk of the message. The extended MD5 makes five passes over each 16-octet chunk.
- ✦ The functions are slightly different in the number of bits in the shifts.
- ✦ MD4 has one constant, which is used for each message word in pass 2, and different constant used for the entire 16 message words in pass 3. No constant is used in pass 1.
- ✦ The extended MD5 uses a different constant for each message word on each pass.

3.1. Algorithm Description

To find the message digest for 'b'-bit message input: The 'b' is an arbitrary non-negative integer; 'b' may be zero; it need not be a multiple of eight, and it may be arbitrarily large. The bits of the message are written down as follows:

m_0 m_1 ... m_{b-1}

Append Padding Bits

The message is to be fed into the message digest computation which must be multiples of 512-bits. The following steps are performed for message padding:

Step 1: The original message is padded by adding a 1-bit, followed by enough 0- bits to leave the message 64-bit less than a multiple of 512-bits.

Step 2: Then a 64-bit quantity representing the number of bits in the unpadded message is appended to the message.

The following figure1 represents the padding process.

	1-512 bits	64 bits
Original Message	**1000...000**	**Original length in bits**

Figure 1: Extended MD5 Figure1: Extended MD5 message padding

The bit order within the octet is a most significant bit to the least significant bit and the octet order is a least significant bit to the most significant bit.

Overview of extended MD5 Message Digest Computation

In extended MD5, the message is processed in 512-bit blocks (sixteen 32-bit words). The following figure2 will illustrate this process. The message digest is a 128-bit quantity (four 32-bit words). Each stage consists of computing a function based on the 512-bit message chunk and the message digest to produce a new intermediate value for the message digest. The value of the message digest is the result of the output of the final block of the message.

Each stage in extended MD5 takes five passes over the message block (as opposed to four for MD5). As with MD4, at the end of the stage, each word of the modified message digest is added to the corresponding pre-stage message digest value. In MD4, before the first stage, the message digest is initialized to $d_0 = 67452301_{16}$, $d_1 = efcdab89_{16}$, $d_2 = 98badcfe_{16}$, and $d_3 = 10325476_{16}$. As with extended MD5, each pass modifies d_0, d_1, d_2, d_3 using m_0, m_1, m_2,...m_{15}. The following five steps are performed to compute the message digest of the message.

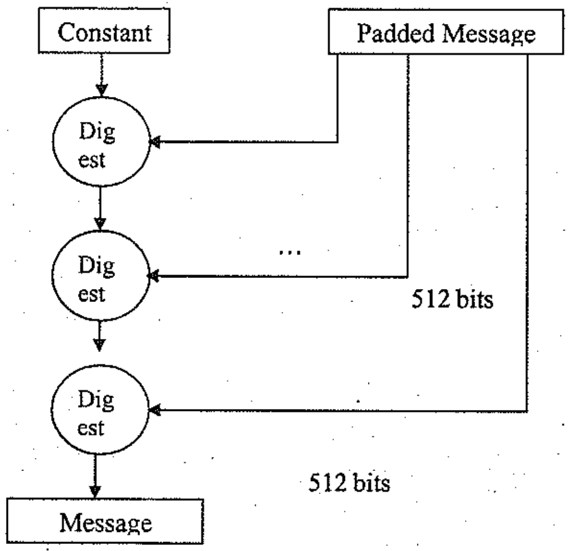

Figure2 : Entire processes for extended MD5

Step 1: Extended MD5 Message Digest Pass 1

For each integer 'i' from 0 through 15,

$d_{(-i)\wedge 3} = d_{(i-i)\wedge 3} + (d_{(-i)\wedge 3} + F(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_i + T_{i+1})$

Where S_1 (i) = 7+5i, so the 'S' cycle over the values 7,12,17,22. This is different 'S_1' from that in MD4. The first few steps of the pass are as follows:

$d_0 = d_1 + (d_0 + F(d_1, d_2, d_3) + m_0 + T_1) = 7$

$d_3 = d_0 + (d_3 + F(d_0, d_1, d_2) + m_1 + T_2) = 12$

$d_2 = d_3 + (d_2 + F(d_3, d_0, d_1) + m_2 + T_3) = 17$

$d_1 = d_2 + (d1 + F(d_2, d_3, d_0) + m_3 + T_4) = 22$

$d_0 = d_1 + (d_0 + F(d_1, d_2, d_3) + m_4 + T_5) = 7$

Step 2: Extended MD5 Message Digest Pass 2

A function G(x, y, z) is defined as $(x \wedge z) \vee (y \wedge \sim z)$. Whereas, the function 'G' is different in extended MD5 to the 'G' function in MD4. A separate step is done for each of the 16 words of the message. For each integer 'i' from 0 through 15,

$d_{(-i)\wedge 3} = d_{(1-i)\wedge 3} + (d_{(-i)\wedge 3} + H(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(3i+5)\wedge 15} + T_{i+33})$

Where S_2 (i) = i(i+7)/2 + 5; so, the 'S' cycle over the values 5, 9, 14, 20. This is a different S_2 from that in MD4. The first few steps of the pass are as follows:

$d_0 = d_1 + (d_0 + G(d_1, d_2, d_3) + m_1 + T_{17}) = 5$

$d_3 = d_0 + (d_3 + G(d_0, d_1, d_2) + m_6 + T_{18}) = 9$

$d_2 = d_3 + (d_2 + G(d_3, d_0, d_1) + m_{11} + T_{19}) = 17$

$d_1 = d_2 + (d_1 + G(d_2, d_3, d_0) + m_0 + T_{20}) = 22$

$d_0 = d_1 + (d_0 + G(d_1, d_2, d_3) + m_5 + T_{21}) = 5$

Step 3: Extended MD5 Message Digest Pass 3

A function H(x, y, z) is defined as $x \oplus y \oplus z$. A separate step is done for each of the 16 words of the message. For each integer 'i' from 0 through 15,

$d_{(-i)\wedge 3} = d_{(1-i)\wedge 3} + (d_{(-i)\wedge 3} + H(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(3i+5)\wedge 15} + T_{i+33})$

Where $S_3(0)=4$, $S_3(1)=11$, $S_3(2)=16$, $S_3(0)=23$; so, the 'S' cycle over the values 4, 11, 16, 23. This is a different S_3 from that of the MD4. The first few steps of the pass are as follows:

$d_0=d_1+(d_0+H(d_1,d_2,d_3)+m_5+T_{33}) = 4$

$d3=d0+(d3+H(d0,d1,d2)+m8+T_{34}) = 11$

$d2=d3+(d2+H(d3,d0,d1)+m11+T_{35}) = 16$

$d1=d2+(d1+H(d2,d3,d0)+m14+T_{36}) = 23$

$d0=d1+(d0+H(d1,d2,d3)+m1+T_{37}) = 4$

Step 4: Extended MD5 Message Digest Pass 4

A function I(x, y, z) is defined as $y \oplus (x \vee \sim z)$. A separate step is done for each of the 16 words of the message. For each integer 'i' from 0 through 15,

$d_{(-i)\wedge 3} = d_{(1-i)\wedge 3} + (d_{(1-i)\wedge 3} + (d_{(-i)\wedge 3} + I(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(7i)\wedge 15} + T_{i+49})$

Where $S_4(i)=(i+3)(i+4)/2$; so, the ⅃s cycle over the 6, 10, 15, 21. The first few steps of the pass are as follows:

$d_0= d_1+(d_0+I(d_1, d_2, d_3)+m_0+T_{49}) = 6$

$d_3= d_0+(d_3+I(d_0, d_1, d_2)+m_7+T_{50}) = 10$

$d_2= d_3+(d_2+I(d_3, d_0, d_1)+m_{14}+T_{51}) = 15$

$d_1= d_2+(d_1+I(d_2, d_3, d_0)+m_5+T_{52}) = 21$

$d_0= d_1+(d_0+I(d_1, d_2, d_3)+m_{12}+T_{53}) = 6$

Step 5: Extended MD5 Message Digest Pass 5

A separate step is done for this phase. The stage 'S_5' is calculated by

$$S_5= ((bit(\sum S_4 \& 63))+1) \oplus bit(\sum S_4)$$

The $\sum S_4$ is calculated by $\sum d_i$ where i = 0 to 15. For example, the $\sum S_4 = 208$ means, the binary value for 208_{10} is 11010000_2 and binary value of 63 is 111111_2.

Step 1: Extract 6-LSB bits from 11010000_2

Step 2: Bit-wise AND operation between 111111_2 and010000_2

Step 3: The result is 010000_2 is added with 000001_2 get 0100012

Step 4: The value 010001_2 is XOR with 010000_2 The output will be 000001_2

Step 5: The output value 000001_2 is converted into decimal value 1_{10} and this value is used as an array index in T_i and the final value got is 'd76aa478'.

The extended hash function uses five passes and provides the better hash values for input messages.

4. Result And Discussion

The extended MD5 for unique hash value has been implemented and the performance has been compared with MD5 and MD4 message digest hashing algorithms. The results were significant in terms of speed and accuracy. The extended MD5 algorithm has been designed and implemented on JAVA under Windows XP operating system using synthetic data. The extended MD5 quality is measured in terms of speed, which differs from algorithm to algorithm.

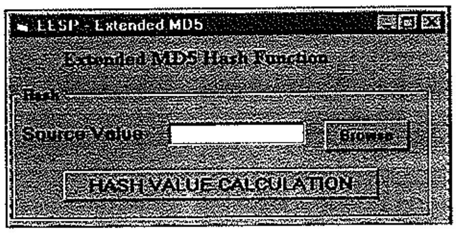

Figure3: Screen view of extended MD5 Algorithm

The above figure3 illustrates the screen view of proposed extended MD5 algorithm. The user enters the two source values or selects the file, which is used as an input for extended MD5 maximum 128-bit with the help of browse button. The hash value calculation button calculating the unique hash value.

The following table1 shows the overall performance results obtained for various runs using the following hardware configuration:

Processor	:	Intel Pentium IV 3.06 GHz
RAM	:	512 MB RAM
Hard Disk	:	80 GB HDD

Table 1: Performance comparison of extended MD5 in terms of time

Run	Extended MD5 (in Seconds)	MD5 (in Seconds)	MD4 (in Seconds)
1	0.001	0.001	0.010
2	0.001	0.003	0.011
3	0.005	0.004	0.018
4	0.004	0.006	0.025
5	0.004	0.009	0.032
6	0.007	0.009	0.037
7	0.008	0.010	0.044
8	0.009	0.003	0.057
9	0.004	0.005	0.062
10	0.002	0.009	0.073

The performance with respect to the hash value calculation time is noted for up to 10 runs for same inputs. This table is used to analyze results about the speed and performance of the proposed extended MD5 algorithm.

It is observed from the above table that the time taken for extended MD5 and MD5 are nearly same for various runs. It also noted that there is a difference of values for certain runs between extended MD5 and MD5. The time difference between Extended MD5 and MD4 is very high. The following table 2 shows the mean and standard deviation of time for Extended MD5 and the MD5.

Table 2: Summary statistics of extended MD5 in terms of time

Statistics Function	ExMD5 (in Seconds)	MD5 (in Seconds)
Mean	0.0045	0.0059
Standard Deviation	0.002799	0.003178

It is observed from the above table2 that the mean of extended MD5 is low and standard deviation of extended MD5 is less than the MD5 and therefore, the extended MD5 is more consistent.

The following figure4 shows the graph of the time for extended MD5, MD5 and MD4 message digest algorithms in hash value calculation.

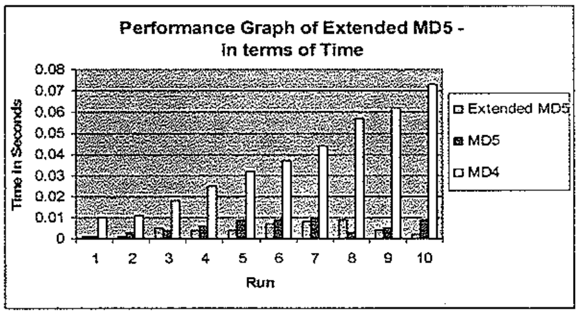

Figure4: Performance Graph of extended MD5 in terms of time

The pictorial representation of the above figure4 clearly shows that the speed of extended MD5 is greater than the other two hashing algorithms. In the accuracy point of view, the proposed extended MD5 has one more pass than MD5 and provides high security than MD5 and MD4.

5. Conclusion

In network communication, the data transmission experiences a great threat by attackers. Even though the attackers cannot read the content of a message, they are capable of changing the content. The economic liability of cyber crimes is also expected to increase two-to-three fold by every year. Most of the cyber crimes are undetected. The security still remains a risky one. The focus of this research paper is mainly on Integrity using extended MD5 hashing algorithm. The extended MD5 hashing algorithm executes five passes and produces unique hash value for each message. This algorithm has been implemented on JAVA using windows XP operating system. The proposed extended MD5 algorithm has been applied in various online transactions for ensure the better integrity than MD5.

6. References

[1] Antoon Bosselaers, *"Even faster hashing on Pentium"*, Proceedings of Eurocrypt'97, Lecture Notes in Computer Science, Springer-Verlag, Vol. 1233, PP 16-17, 1997.

[2] Jie Liang, and Xue-jie lai, *"Improved collision attack on hash function MD5"*, Journal of Science and Technology, Springer-Verlag, Vol. 22, No. 1, PP 79-87, 2007.

[3] Jose L Munoz, Jordi Forne, Oscar Esparza, and Miguel Soriano, *"Certificate revocation system implementation based on the Mrekle hash tree"*, International Journal of Information Security, Springer-Verlag, Vol. 2, No. 2, PP 110-124, 2004.

[4] Karl Krukow, and Mogens Nielsen, *"Trust structures – denotational and operational semantics"*, International Journal of Information Security, Springer-Verlag, Vol. 6, No. 3, PP 153-181, 2007.

[5] Lars R. Knudsen, Xuejia Lai, and Bart Preneel, *"Attacks on fast double block length hash functions"*, Journal of Cryptology, Springer-Verlag, Vol.11, No.1, PP 59-72, 1998.

[6] Mihir Bellare, Ran Canetti and Hugo Krawczyk, *"Keying hash functions for message authentication"*, Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology, Lecture Notes in Computer Science, Vol. 1109, PP 1-15, 1996.

[7] Phong Q. Nguyen, and Igor E. Shparlinski, *"The insecurity of digital signature algorithms with partially known nonces"*, Journal of Cryptology, Springer-Verlag, Vol. 15, No. 3, PP 151-176, 2002.

[8] Rose Anderson and Eli Biham, *"Tiger: A fast new hash function"*, citeseer.ist.psu.edu/anderson96tiger.html, PP 1-13, 1996.

[9] Rose Anderson and Math .C, *"The classifications of hash functions"*, citeseer.ist.psu.edu/33025.html, PP 1-11, 1993.

[10] Tinjie Cao, Dongdai Lin, and Rui Xue, *"A randomized RSA-based blind signature scheme for electronic cash"*, Computers and Security, Elsevier, Vol. 24, No. 1, PP 44-49, 2005.

[11] Wen-Ai Jackson, Keith M. Martin, and Christine M. O'Keefe, *"Mutually trusted authority-free secret sharing schemes"*, Journal of Design Codes and Cryptography, Springer-Verlag, Vol. 10, No. 4, PP 261-289, 1997.

[12] Yacine Challal, Abdelmadjid Bouabdallah, and Hatem Bettahar, *"Hybrid hash chaining scheme for adaptive multicast source authentication of media-streaming"*, Computers and Security, Elsevier, Vol. 24, No. 1, PP 57-68, 2005.

[13] Theodosios Tsiakis, and George Sthephanides, *"The concept of security and trust in electronic payment systems"*, Computers and Security, Elsevier, Vol. 24, No. 1, PP 10-15, 2005.

[14] Thomas Wu, *"The secure remote password protocol"*, IEEE Journal on Selected Areas in Communication, IEEE Press, Vol. 11, No. 5, PP 648-656, 1993.

[15] Tian-Fu Lee, Chai-Chaw Chang, and Tzonelih Hwang, *"Private authentication techniques for global mobility networks"*, Journal of Wireless Personal Communications, Springer-Verlag, Vol. 35, No. 4, PP 329-336, 2005.

[16] Stephanie Alt, *"Authenticated hybrid encryption for multiple recipients"*, IEEE Journal on Selected Areas in Communications, IEEE-Press, Vol.11, No.5, PP 156-182, 2006.

[17] Steven M Bellovin, and Micheal Merritt, *"Encrypted key exchange: password-based protocols secure against dictionary attacks"*, Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, PP 48-56, 1992.

Author's Biography

Dr. S. Karthikeyan received the Doctorate Degree in Computer Science and Engineering from the Alagappa University, Karaikudi in 2008. He is currently working as a Reader and Head in Department of Computer Science, Karpagam Arts and Science College (Autonomous), Coimbatore. His research interests include Network Security using Cryptography.