ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V2I2M13-032012

VOLUME 2 ISSUE 2 March 2012

A PRIORITY BASED COLLISION AVOIDANCE ALGORITHM FOR SECURE DATA EXCHANGE

Harpreet Kaur
Research Scholar
Department of Computer Science
Lovely Professional University
Jallandhar, India

Abstract

Most of the time information handled by organizations has been collected and processed by computers and transmitted across networks to other computers. How can we protect this information from unauthorized access? The secure transmission of data in transit relies on both cryptography and authentication − on both the hiding or concealment of the data itself, and on ensuring that the computers at each end are the computers they say they are. In this paper a priority based collision avoidance algorithm for secure data exchange is proposed where the encryption and decryption process uses hash function and authentication is supported via username password. The algorithm provides the unique feature of collision and redundancy avoidance and also supports the priority based response from receiver of data or message.

Keywords: *Priority, collision, data security, authentication, cryptography.*

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V2I2M13-032012

VOLUME 2 ISSUE 2 March 2012

1 INTRODUCTION

Data security has become one of the most important concerns for governments, financial institutions, hospitals, and private businesses. An important security risk is that information can be captured and read during its transmission. How do we protect this information from being read by intruders? The secure transmission of data in transit relies on both cryptography and authentication − on both the hiding or concealment of the data itself, and on ensuring that the computers at each end are the computers they say they are.

Cryptography, to most people, is concerned with keeping communications private. Encryption is the transformation of data into some unreadable form. Its purpose is to ensure privacy by keeping the information hidden from anyone for whom it is not intended. Decryption is the reverse of encryption; it is the transformation of encrypted data back into some intelligible form. Encryption and decryption require the use of some secret information, usually referred to as a key. The data to be encrypted is called as plain text. The encrypted data obtained as a result of encryption process is called as cipher text. Depending on the encryption mechanism used, the same key might be used for both encryption and decryption, while for other mechanisms, the keys used for encryption and decryption might be different. There are several ways of classifying cryptographic algorithms. In general they are categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use as in [1]. The three types of algorithms are depicted as follows

1) Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption. The most common algorithms in use include Data Encryption Standard (DES), Advanced Encryption Standard (AES).

2) Public Key Cryptography (PKC): Uses one key for encryption and another for decryption. RSA (Rivest, Shamir, Adleman) algorithm is an example.

3) Hash Functions: Uses a mathematical transformation to Irreversibly "encrypt" information. MD (Message Digest)

In this paper a priority based collision avoidance algorithm for secure data exchange is proposed where the encryption and decryption process uses hash function and authentication is supported via username password. The algorithm provides the unique feature of collision and redundancy avoidance and also supports the priority based response from receiver of data or message. Section II presents the detailed study of hash function. Section III describes the proposed algorithm for secure data exchange. Finally section IV concludes the paper and presents the future work directions.

2. Basic Definitions, Properties, Classification and Requirements of Hash Functions

Hash functions have been used in vast variety of cryptographic application and must provide different security properties depending on the security requirements of the application. The well known basic security properties of hash functions are preimage resistance, second preimage resistance and collision resistance. They are explained below:

• Preimage resistance: for any given code h, it is computationally infeasible to find x such that $H(x) = h$.

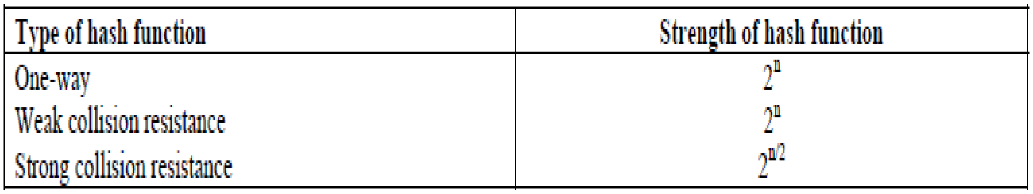
• Second preimage resistance: for any given input x, it is computationally infeasible to find y $\neq x$ with $H(y) = H(x)$.

• Collision resistance: it is computationally infeasible to find any pair (x, y) such that H(y) = H(x).

Properties preimage resistance, second preimage resistance and collision resistance are also known as one-way, weak collision resistance and strong collision resistance respectively. Table 1 summarizes the level of effort required producing a birthday or square root attack for different types of hash functions, assuming n-bit result [4].

Cryptographic hash function can be traditionally classified as unkeyed hash functions and keyed hash functions. Unkeyed hash functions, also known as modification detection codes (MDCs), use message as a single input whereas keyed hash functions, also known as message authentication codes (MACs), can be viewed as hash functions which take two functionally distinct inputs, a message and a secret key. Unkeyed hash function is further classified into one-way hash function (OWHF), collision resistant hash function (CRHF), universal one way hash function (UOWHF) [2, 8, 11, 13].The construction of CRHF is hard than OWHF. CRHF usually deals with longer length hash values.

Table 1: Strength of different hash functions



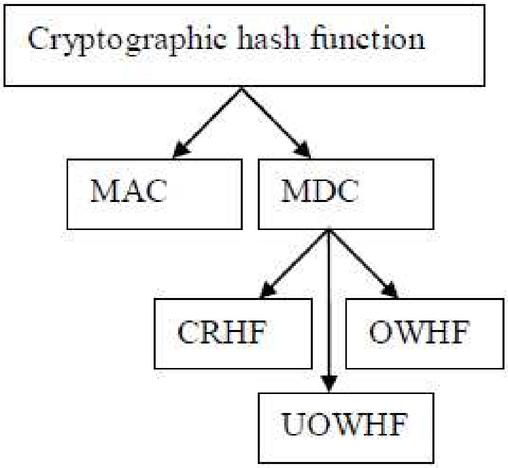
Type of hash function	Strength of hash function
One-way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

2.1 Unkeyed Hash Function

An unkeyed hash function is a function $h:\{0,1\}*\rightarrow\{0,1\}n$, for a fixed positive integer n which has, as a minimum, the following two properties:

- Compression: h maps an input x of arbitrary finite bit length, to an output h(x) of fixed bit length n.
- Ease of computation: given h and an input x, h(x) is easy to compute.

Figure 1: Classification of cryptographic hash function



2.1.1. One-way Hash Function (OWHF)

One-way hash function is a hash function with properties preimage resistance and second preimage resistance. For these, finding an input which hashes to a prespecified hash value is difficult.

2.1.2. Collision Resistant Hash Function (CRHF)

A collision resistant hash function is a hash function with properties second preimage resistance and collision resistance. For these, finding any two inputs having the same hash value is difficult.

2.1.3. Universal One-way Hash Function (UOWHF)

In a universal one-way hash function, for randomly chosen input x, key k and the function hk, it is hard to find $y \neq x$ such that $hk(x) = hk(y)$

2.2. Keyed Hash Function (MAC)

A keyed hash function is a function $hk: \{0,1\}k \times \{0,1\}^* \rightarrow \{0,1\}n$ for fixed positive integer n and k, if it satisfies following two properties:

- Compression: hk maps an input x of arbitrary finite bit length, to an output $hk(x)$ of fixed bit length n

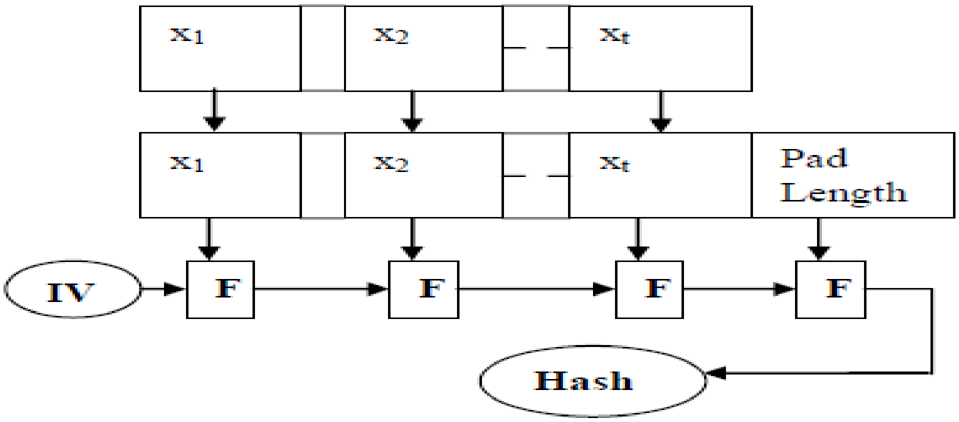
- Ease of computation: for a known function hk, given a value k and an input x, $hk(x)$ is easy to compute. The result is called MAC value.

• Computation-resistance: given zero or more text-MAC pairs (xi, hk(xi)), it is computationally infeasible to compute any text-MAC pairs(x, hk(x)) for any new input x ≠ xi.

Almost all hash functions are iterative processes which hash inputs of arbitrary length by processing successive fixed size blocks of the input. The input X of arbitrary finite length is divided into fixed length t-bit blocks, x1 through xt. This number of fixed length blocks must be multiple of the block length for attaining the overall bit length, it typically involves appending extra bits (padding). Hash function can be described as the following: H0 = IV, Hi = F (xi,Hi-1), i=1,2……..t; h(x)=Ht where IV is stood for initial value and the result of hash function F is called the hash round function. Such a recursive construction known as Merkle-Damgård hash construction designed by Ralph Merkle and Ivan Damgård independently in 1989[10,46]

Apart from the classification of keyed and unkeyed hash functions, they can be classified into other ways such as hash function based on block cipher, hash function based on modular arithmetic and dedicated hash functions. We are giving a brief review of these hash functions.

Figure 2: Merkle-Damgård Construction



2.3. Hash Function Based on Modular Arithmetic

Number theory problems are used to design these hash functions. Security of such hash function is directly proportional to the hardness of these problems. The two most important cryptosystems, based on modular arithmetic are RSA public key cryptosystem and ElGamal cryptosystem. Hash functions that are based on modular arithmetic can have variable digest length, depending on the size of modulus. Example of this approach is MASH-1(Modular Arithmetic Secure Hash algorithm-1). The purpose of employing modular arithmetic is to save on implementation costs. A cryptographic hash function can use modular arithmetic as the basis of it's compression function. This allows the reuse of existing implementation of modular arithmetic. An advantage of these schemes is that it is easy to scale the security level by choosing a modulus of appropriate length. A significant disadvantage is that hash functions based on modular arithmetic are very slow, even when compared to block cipher based construction.

2.4. Hash Function Based on Block Cipher

There have been many efforts to construct hash functions from the existing block ciphers. The main motivation to construct a hash function based on a block cipher is the minimization of design and implementation effort. The advantage of this approach is that the trust in the security of block ciphers can be transformed on to the hash functions. Hash functions developed using block ciphers are either keyed or MDCs. Hash functions based on block ciphers are usually slower when compared to that of the dedicated hash functions. Davies-Meyer, Miyaguchi-Preneel, Matyas-Meyer-Oseas, MDC-2 and MDC-4 are some methods to generate a compression function of a hash function from a block cipher.

2.4.1. Davies-Meyer (DM) Scheme

The DM-scheme was proposed independently by Davies and by Meyer. This scheme can be used with any block cipher. The message block Mi, that is hashed in each step of this scheme has length l equal to the key length k of the block cipher, i.e., l = k. The block cipher E takes a block of the message Mi as a key and Hi-1 the previous hash value as a plain text to be encrypted. The output of the cipher text is then XORed with the previous hash value Hi-1 to produce the next hash value Hi.

$$Hi = EMi\ (Hi-1)\ \square\ Hi-1$$

Figure 3: DM scheme

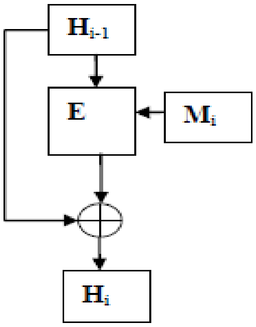


Figure 4: MMO scheme

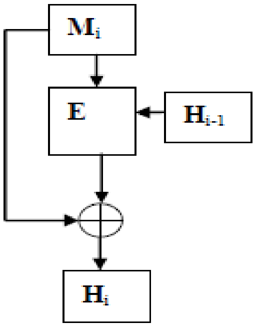
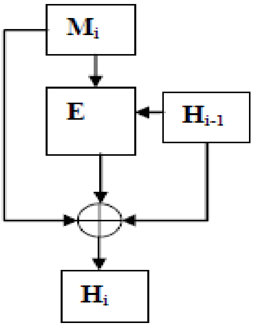


Figure 5: Miyaguchi-Preneel scheme



2.4.2. Matyas-Meyer-Oseas (MMO) Scheme

In this construction current message block is encrypted. In encryption previous hash value Hi-1 is used as a key. Then this encrypted message block is XORed with current message block Mi to produce hash value Hi. This scheme constructs the compression function as follows:

$$H_i = E_{Hi\text{-}1}(M_i) \oplus M_i$$

2.4.3. Miyaguchi-Preneel Scheme

This scheme is an extended version of MMO scheme. The only difference is that, in this scheme the previous hash value Hi-1 is also XORed with the cipher text along with the message block Mi.

$$H_i = E_{Hi\text{-}1}(M_i) \oplus M_i \oplus H_{i\text{-}1}$$

2.4.4. MDC-2 and MDC-4 Scheme

MDC-2 and MDC-4 are manipulation detected codes requiring 2 and 4, respectively, block cipher operations per block of hash input.MDC-2 scheme was originally defined for use with the DES block cipher; however it can be instantiated with any block cipher. The MDC-2 compression function contains two parallel block cipher encryptions and can be seen as a two-way parallel extension of the MMO scheme.MDC-4 employ a combination of four iteration of Matyas-Meyer-Oseas method to generate a double length hash [2, 11, 12].

2.5. Dedicated Hash Function

Dedicated hash functions are specially designed from the scratch for the purpose of hashing a plain text with optimized performance and without being constrained to reusing existing system components such as block ciphers and modular arithmetic. These hash functions are not based on hard problems such as factorization and discrete logarithms. The most popular method of designing compression functions of dedicated hash functions is a serial successive iteration of a small step function. MD2[32],MD-4[17],MD-5[18],SHA-1[25],SHA-2[15],TIGER[41], RIPEMD[19] and RIPEMD-160[20] are some examples of dedicated hash functions. Almost all the dedicated hash functions are based on the basic construction of Merkle-Damgård.

3. Proposed Algorithm and Data Flow Diagram

In this section a priority based collision avoidance algorithm for secure data exchange is proposed which provides the security to the long message in terms of authentication and confidentiality so that the privacy of data does not breach and not any unauthorized user access the data. This algorithm avoids the collision and storage of repeated data when similar messages repeated over network at times and also provides the facility to receiver for responding according to the priority wise and separately to each sender very efficiently. The functional flow diagram and algorithm steps are described in figure 6 and 7 respectively

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V2I2M13-032012

VOLUME 2 ISSUE 2 March 2012

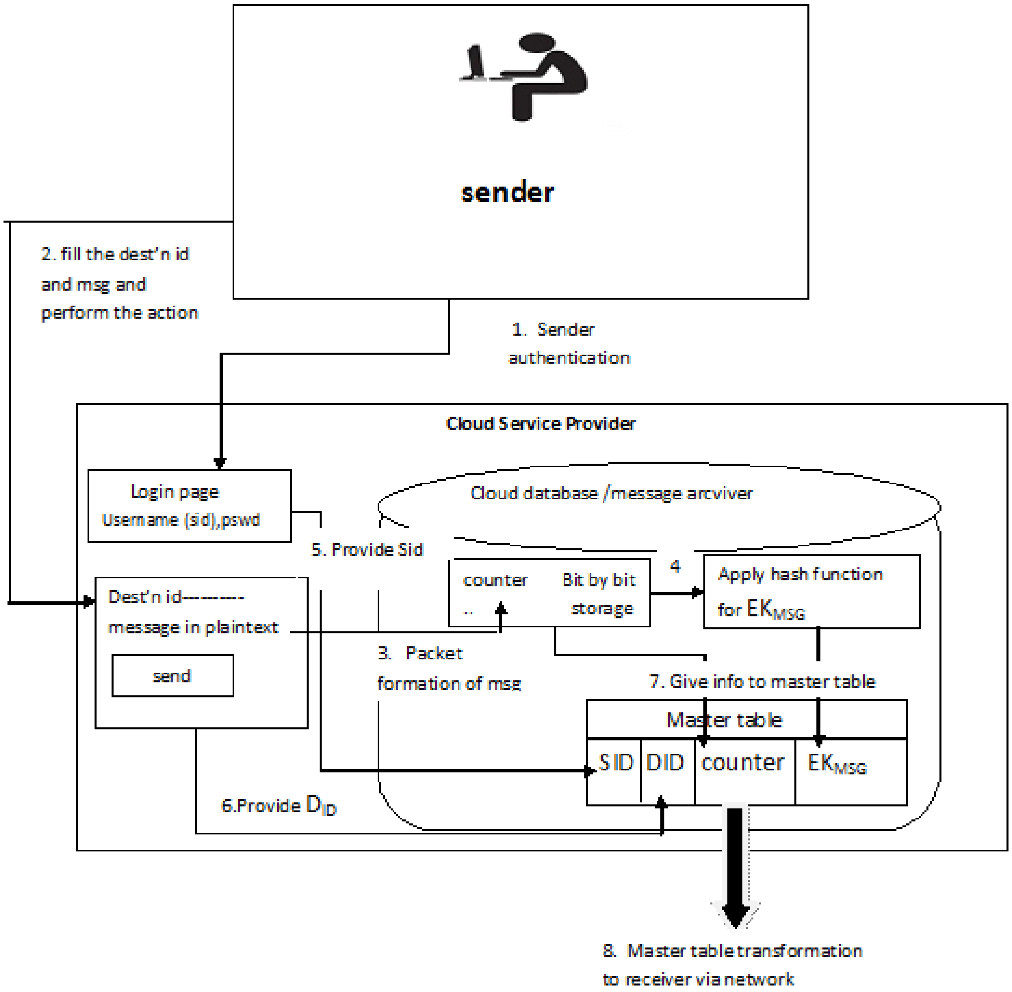

sender
2. fill the dest'n id and msg and perform the action
1. Sender authentication
Cloud Service Provider
Login page
Username (sid),pswd
Cloud database /message arcviver
5. Provide Sid
counter
..
Bit by bit storage
4
Apply hash function for EK_{MSG}
Dest'n id---------
message in plaintext
send
3. Packet formation of msg
7. Give info to master table
Master table
SID
DID
counter
EK_{MSG}
6.Provide D_{ID}
8. Master table transformation to receiver via network

Figure 6. Functional flow diagram for sending encrypted message

Algorithm for sending encrypted message or data

```
1. Authentication of sender
   (a) Sender login to cloud server where:
        $S_{ID}$ ← Username   /* set username of sender as sender id $S_{ID}$*/
        Psw ← password
2. Message Sending: During message sending the following steps are followed by sender:
   (a) DID ← Destination user name
       /* sender filled the destination Username to which he wants to send the message that is stored as destination id DID*/
   (b) Msg ← Plain text   /* sender write the message in to the plain text form*/
       And send to the receiver
3. Storage of encrypted data in to the cloud server database : After sending the data the message archiver present into the cloud server data base performs the following actions:
   (a) Packet formation :
       (i) Message archiver receives the message and store it as one by one bit or
              character in message field of temporary created table
       (ii) For each bit counter ←counter+1
   (b) Encryption: message archiver perform the hash function on message field
       $EK_{MSG}$ ← Hash(message field)
   (c) Storage of Master table: message archiver stores the master table into the cloud database which involves following fields:
       Master Table ← ($S_{ID}$,$D_{ID}$, $EK_{MSG}$, Sequence no/counter)
4. The master table is sent to the receiver via communication link.
5. End of algorithm
```

The brief description of algorithm is as follow

Steps For sending encrypted data

Module 1: Authentication: The sender who wants to send the message firstly login at cloud server for authentication purpose where the sender username is taken as sender id.

Module 2: Message Sending: After login process the CSP requests to the sender to fill the information of destination identity and the message in the form of plain text that he wants to send to the receiver. After providing the required information to CSP the sender performs the action of sending the message by clicking on send button.

Module 3: Packet formation and encryption of message: Before sending the message from sender to receiver the CSP firstly converted the messages into packets and stores it as one by one bit. The counter is also incremented for each character/ bit of message which also deals with the problem of the collision of similar words caused due to the repetition of similar words during the transmission of long message. Cloud service provider also performs the hash function on message for encryption and maintaining their confidentiality.

Module 4: Storage of data as Master Table: The message archiver maintains the master table and stores it into Cloud database that has sender id, destination id, encrypted message and sequence number of whole message in its fields. The master table is then sent to the receiver through any of the network mediums.

The functional flow diagram and algorithm steps are described in figure 8 and 9 respectively.

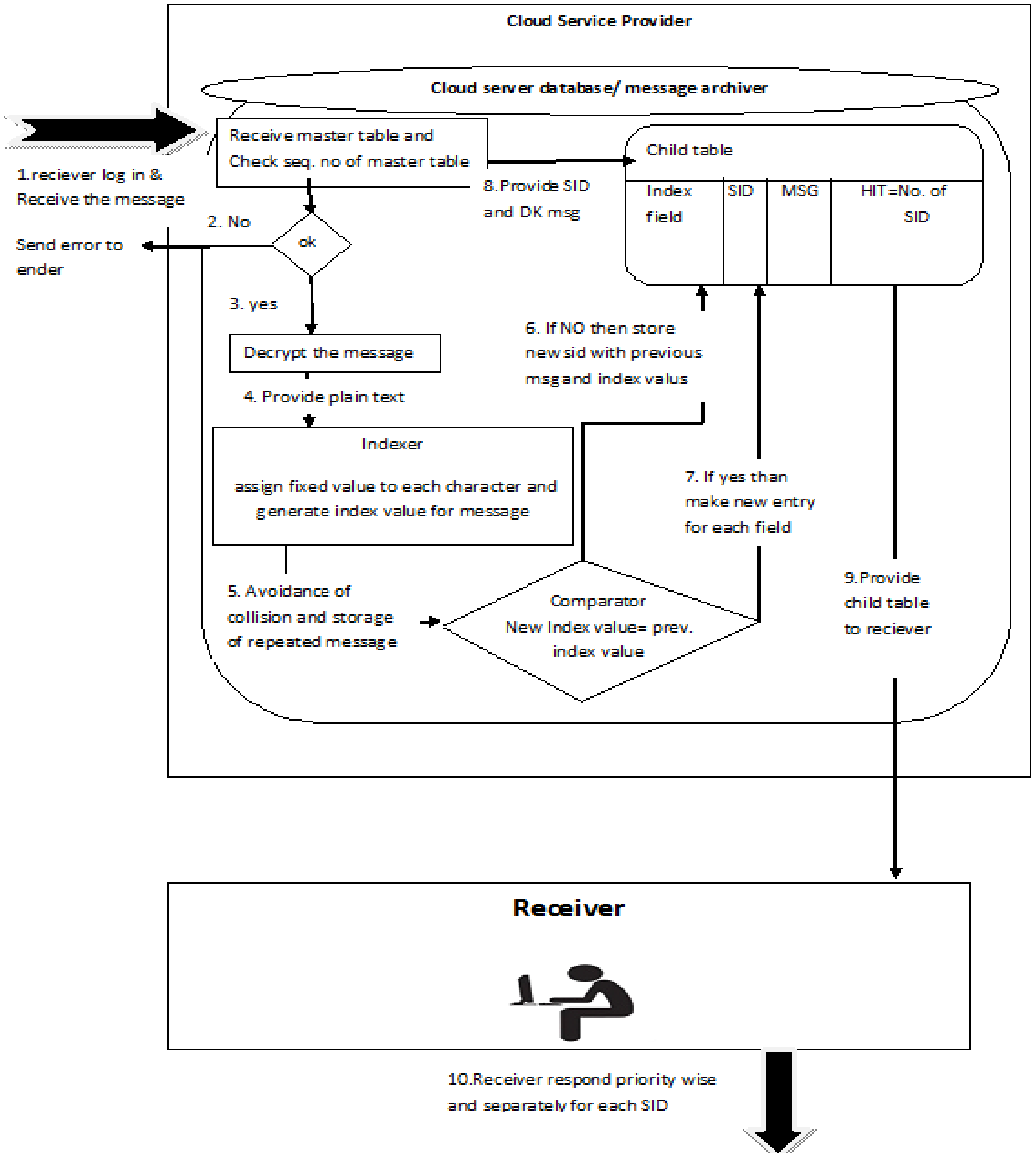

Cloud Service Provider
Cloud server database/ message archiver
1.reciever log in &
Receive the message
Receive master table and
Check seq. no of master table
Child table
Index field
SID
MSG
HIT=No. of SID
8.Provide SID
and DK msg
2. No
ok
Send error to
ender
3. yes
Decrypt the message
6. If NO then store
new sid with previous
msg and index valus
4. Provide plain text
Indexer
assign fixed value to each character and
generate index value for message
7. If yes than
make new entry
for each field
5. Avoidance of
collision and storage
of repeated message
Comparator
New Index value= prev.
index value
9.Provide
child table
to reciever
Receiver
10.Receiver respond priority wise
and separately for each SID

Figure 9. Functional Flow diagram for decrypting the message at receiver side

ALGORITHM FOR DECRYPTION OF MESSAGE AND COLLISION AVOIDANCE AT RECEIVER SIDE

```
1. Receiving and decryption of incoming message
   (a) Receiver side message archiver receives the master table
   (b) Check for sequence number
          If
             sequence number is ok
          Then
                go to step 2
        otherwise
                      send an error message to sender
2. Decrypt the message:   $DK_{MSG} \leftarrow Hash_{REVERSE}(EK_{MSG})$
3. Creation of child table into the cloud database
   (a) Assigned the fixed value for each character of decrypting message i.e.
       $FX_i \leftarrow i$   /* for each character I the fixed value is $FX_i$ where i= 1 to n */
   (b) Index value $\leftarrow$ $(FX_1 . FX_2 . FX_3 ......... FX_4)$
       /* where the index value is equal to the concatenation of fixed values of all
       characters of decrypted msg */
   (c) Message archiver stores the child table into the cloud database as follow:
       Child table $\leftarrow$ (index value, $S_{ID}$, $DK_{MSG}$,HITS)
4. Avoidance of collision and storage of repeated message:
   (a) For each incoming message
```

```
If
    Index value $_{new}$=index value $_{prv}$
                    /*where message archiver match the index value for new
                    decrypting message with the index values of previous
                    messages existed in child table */
Then
    Store new SID with the previous matched index value and
        HIT ← HIT+1
Otherwise
    make a new entry for each field of child table
```

5. Receiver's response to the sender:
   (a) Receiver responds priority wise and separately for each S_{ID} existed into the child table where greater the number of hits represents the higher the priority of message.
6. End of algorithm.

Steps for decrypting the message and avoidance of storage of repeated data at receiver side

Module 1: Check the sequence number for message: receiver side buffer or message archiver receives the Master table and check the sequence number of message. If the sequence number is correct then message archiver decrypts the encrypted message but is the sequence number is not correct then it sends the error message to sender.

Module 2: creation of child table: The message archiver assigns the fixed value for each character of decrypted message i.e. assign FXi for each i where i=1to n. Message archiver also creates the index value for each message which is the concatenation of the fixed values that are assigned to each character of message and then store the child table into the cloud server that has following fields: index value for each message sender ids, decrypted message and hits that is incremented every time when a sender id is stored in child table.

Module 3: Avoidance of collision (loss of information) and storage of repeated messages: for each incoming message , message archiver always checks weather the index value of new arrived message is equal to the index value of any previously existed message. If it matches then only the sender id is stored in child table with the previously existed index value which prevents from the storage of same repeated message again in child table. But if the index values are not matched then message archiver saves a new entry for each field of child table. This technique also prevents from the loss of information due to the collision of same message at times because it always saves the sender id and also increment the number of hits whenever a new message arrived at receiver side.

Module 4: Priority wise response of receiver: The child table is sent to the receiver and receiver responds priority wise and separately for each sender id present into the child table. Where greater the number of hits represents highest the priority of message.

4. Conclusion and future work

This paper presented priority based collision avoidance algorithm for secure data exchane. The security solutions are mainly considered the authentication and cryptography process. The collision and redundancy are also avoided with the help of sequence number and child table created in the proposed algorithm. Future extension will Provide the detailed evaluation of algorithm implementation.

References

[1] S.Bakhtiari, R. Safavi-Naini and J. Pieprzyk, 1995."Cryptographic hash functions: A Survey", Technical Report 95-09, Department of Computer Science, University of Wollongong.

[2] A.J. Menezes, P.C. Van Oorschot, S.A.Vanstone Handbook of Applied Cryptography, CRCpress, 1996.

[3] RSA Laboratories frequently asked questions about today's cryptography, version 4.1.2000. Available: http://www.rsasecurity.com.

[4] P. Rogaway and T. Shrimpton, 2004."Cryptographic hash-function basics: Definitions, implications and separations for preimage resistance, second-preimage resistance, and collision resistance", FSE 2004.

[5] D.R. Stinson, 1994. "Universal hashing and authentication codes." Designs, Codes and Cryptography, 4, pp. 369–380.

ISSN (Online) 2249 - 054 X

International Journal of Computing and Corporate Research

Specialized and Refereed Journal for
Research Scholars, Academicians, Engineers and Scientists

http://www.ijccr.com

International Manuscript ID : ISSN2249054X-V2I2M13-032012

VOLUME 2 ISSUE 2 March 2012

[6] D.R. Stinson, 2006.” Some observations on the theory of cryptographic hash functions” Designs, Codes and Cryptography, 38(2), pp. 259–277.

[7] Ilya Mironov, 2005. “Hash functions: Theory, attacks, and applications”, J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp. 68–73.

[8] Nigel Smart, Cryptography: An Introduction. McGraw-Hill,Third edition,2003.Available: http://www.cs.bris.ac.uk/~nigel/Crypto_Book/

[9] I. Damgård, 1987. “Collision free hash functions and public key signature schemes”, in: Proc. of Eurocrypt-87, in LNCS, vol. 304, pp. 203-216.

[10] I.B.Damgård, 1989 “A design principle for hash functions”. In Gilles Brassard, editor, Advances in Cryptology: CRYPTO 89, volume 435 of Lecture Notes in Computer Science, pp. 416-427.

[11] B.Preenel, 1994.”Cryptographic hash functions”, Transactions on Telecommunications, VOL5, pp. 431-448.

[12] Bart Preneel, 1993.”Analysis and Design of Cryptographic Hash Functions”, Dissertation, Katholieke Universiteit Leuven.

[13] William Stallings. Cryptography and Network Security: Principles and Practice. Third edition, Prentice Hall. 2003.

[14] W.Diffie and M.E Hellman, 1976. “New directions in cryptography”, IEEE Transaction on Information Theory. vIT-22 i6, pp. 644-654.

[15] NIST,2002, “Secure Hash Standars“,FIPS PUB 180-2.

[16] X. Wang, X. D. Feng, X. Lai and H.Yu, 2004. “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD”, rump session, CRYPTO 2004.

[17] R.L.Rivest, 1992.”The MD4 Message Digest Algorithm”,RFC 1320.

[18] R.L.Rivest, 1992.”The MD5 Message Digest Algorithm”,RFC 1321.

[19] RIPEMD, Research and Development in Advanced Communication Technologies in Europe, RIPE Integrity Primitives: Final Report of RACE Integrity Primitives Evaluation (R1040), RACE, June 1992.

[20] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel, 1996.” RIPEMD-160– A Strengthened Version of RIPEMD”, Lecture Notes on Computer Science, Volume 1039, Fast Software Encryption 1996, pp. 71–82.

[21] X. Wang, H. Yu and Y.L. Yin, 2005. “Efficient Colision Search Attacks on SHA-0”, CRYPTO 2005.

[22] XiaoyunWang, Yiqun Lisa Yin, and Hongbo Yu, 2005.”Finding Collisions in the Full SHA-1, Lecture Notes in Computer Science, Volume 3621, Advances in Cryptology – CRYPTO 2005 Proceedings, pp. 17–36.

[23] Xiaoyun Wang, Andrew Yao, and Frances Yao, 2005.”New Collision Search for SHA-1, Presented at rump session of CRYPTO 2005.

[24] K. Matusiewicz and J. Pieprzyk, 2006. “Finding good differential patterns for attacks on SHA- 1”, Lecture Notes in Computer Science, Volume 3969, pp. 164-177.

[25] NIST, “Secure Hash Standar“, 1995. FIPS PUB 180-1.

[26] Florent Chabaud, Antoine Joux, 1998. “Differential collisions in SHA-0,” Advances in Cryptology-CRYPTO’98.

[27] Eli Biham and Rafi Chen, 2004.”Near-Collisions of SHA-0”, Lecture Notes in Computer Science, Volume 3152, Advances in Cryptology – Crypto 2004 Proceedings, pp. 290–305.

[28] Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, William Jalby, 2005. “Collision in SHA-0 and Reduced SHA-1”, Advances in Cryptology-EUROCRYPT 2005.

[29] Vincent Rijmen and Elisabeth Oswald, 2005.” Update on SHA-1”. In Alfred Menezes, editor, Topics in Cryptology - CT-RSA 2005, The Cryptographers’ Track at the RSA Conference 2005, San Francisco, CA, USA, volume 3376 of LNCS, pp. 58–71.

[30] Christophe De Canniẻre, Florian Mendel, and Christian Rechberger, 2007. “ Collisions for 70- Step SHA-1, On the Full Cost of Collision Search” , In Selected Areas in Cryptography, pp. 56- 73.

[31] Christophe De Canniẻre and Christian Rechberger,2008. “Preimages for Reduced SHA-0 and SHA-1”, In CRYPTO 2008, pp. 179-202.

[32] R.L.Rivest, 1992. “The MD2 Message-Digest Algorithm”, RFC 1319.

[33] Bert den Boer and Antoon Bosselaers, 1991. “An Attack on the Last Two Rounds of MD4”, Lecture Notes in Computer Science, Volume 576, Advances in Cryptology – Crypto ’1991 Proceedings, pp. 194–203.

[34] Hans Dobbertin, 1996. ”Cryptanalysis of MD4”, Lecture Notes in Computer Science, Volume 1039, FSE 1996, pp. 53–69, February 1996.

[35] Hans Dobbertin, 1997. ”The First Two Rounds of MD4 are Not One-Way”, Lecture Notes in Computer Science, Volume 1372, FSE 1998, pp. 284–292.

[36] Bert Den Boer and Antoon Bosselaers, 1994. „Collisions for the Compression Function of MD5”, Advances in Cryptology, Proceedings Eurocrypt ’93, Springer-Verlag LNCS 765, pp. 293–304.

[37] Hans Dobbertin, 1996. “Cryptanalysis of MD5”, Rump Session, EUROCRYPT 1996.

[38] Vlastimil Klima, 2006. “Tunnels in Hash Functions: MD5 Collisions Within a Minute.”,Cryptology ePrint Archive, Report 2006/105, 2006.Available: http://eprint.iacr.org/.

[39] Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta, 2005.” Improved Collision Attack on MD4”, Cryptology ePrint Archive, Report 2005/151, May 2005. http://eprint.iacr.org/2005/151.pdf

[40] Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry, 1993.” HAVAL – A One-Way Hashing Algorithm with Variable Length of Output”, Lecture Notes in Computer Science, Volume 718, Advances in Cryptology – Auscrypt ’92, pp. 83–104.

[41] R.J.Anderson,.E.Biham., 1996. “TIGER: A Fast New Hash Function”,FSE, LNCS, vol. 1039, pp. 89–97.

[42] Paulo S.L.M. Barreto and Vincent Rijmen ,2000.” The Whirlpool Hash Function “, First open NESSIE Workshop.

[43] Eli Biham and Orr Dunkelman, 2006. “A framework for iterative hash functions-HAIFA”, NIST Second Hash Functions Work Shop, Santa Barbara.

[44] D. Hong, S. Jaechul, S. Hong, S. Lee and D. Moon, 2005. “A new dedicated 256-bit hash function: FORK-256”. First NIST Workshop on Hash Functions.

[45] H. Gilbert and H. Hanschuh, SAC 2003,“Security Analysis of SHA-256 and sisters, Selected Areas in Cryptography”, Ottawa, Canada, Lecture Notes in Computer Science, vol. 3006, M. Matsui and R. Zuccheratopp (Eds), Springer,2004, pp. 175-193.

[46] R. Merkle, 1989.” One way hash functions and DES. In: Brassard, CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg.

[47] H. Tiwari and K. Asawa, 2010, “A Secure Hash Function MD-192 with Modified Message Expansion”, IJCSIS, Vol. 7, No. 2, pp. 108-111.

[48] C. S. Jutla and A. C. Patthak, 2005. “A simple and provable good code for SHA message expansion”. In IACR ePrint archive 2005/247.