International Journal of Engineering Applied Sciences and Technology, 2019
Vol. 4, Issue 8, ISSN No. 2455-2143, Pages 265-270
Published Online December 2019 in IJEAST (http://www.ijeast.com)

MULTI SIGNATURE SCHEME FOR CONSISTENCY IN WEB SERVICES

Isha Narula
Department Of Computer Science
Chitkara University, Rajpura, Punjab

Jahanvi Mangla
Department Of Computer Science
Chitkara University, Rajpura, Punjab

Abstract— **The most challenging task in the web arena is the delivery of the payloads in the wireless environment because these payloads may be corrupted at any point of time. The web services are generally dealing with small chunks of payloads and they need fast transactions with minimal delay where hundred percentage accuracy is compulsory. The existing standards of the symmetric and asymmetric encryption although provide confidentiality and authentication [4] but they are still unable to provide data integrity and consistency. The light weight data like JavaScript Object Notation (JSON) and the eXtensible Markup Language (XML) [1] were introduced in order to make the data access easier and faster. These objects follow the existing methods of generating message digests [7] but still they need a special kind of message digest [7] to ensure the data integrity at any level. In this paper, a new approach of generating distributed message digests [7] for the XML [1] and JSON files is introduced to enhance and optimize the error checking and correcting codes. The foremost target of this paper is to situate the precise errors for resending rather than sending full payload for even a small packet error.Applying the concourse brilliance and flea outpost principium it has been proven that different digests can be generated for a single document and also cater extra sphere for the advancement of more powerful and efficient approach towards distributed message digests. The extant algorithms do not unlock all the ways of error correction and detection completely. As of today no technique is efficient enough to cover all the problems and to find its elucidation. There is the need to research for new and efficient schemes for data consistency and security. Dynamic and scalable algorithms flexible for different files which can secure the data are demanded. The various algorithms are explained and there is also an inauguration of new parameters which can improve data consistency and integrity. This can spark to incitement of readers furthermore researchers to evaluate and establish the new ways of data consistency. This paper brings better perception of various error checking and correcting ways of XML [1] and JSON files and cater better results for the use of research technique in the development of data integrity and consistency.**

Keywords— XML, JSON, Message Digests, Symmetric and Asymmetric Encryption, Distributed Message Digest

I. INTRODUCTION

XML- XML [1] is a platform independent and self-descriptive meta language contrived for concocting, exemplification and exhibition of text. This terminology specifies cipher for formatting, both the blueprint and design within the text file. The cipher for specifying formatting are known as tags. This terminology also defines protocols for concealing documents in a format that can be read by an individual and also system parable. Validation of XML [1] document can take place in two different ways. The first method is Document Type Definition (DTD) [2]. A DTD [2] is a clearly defined record or markup proclamation specifying description or set of laws of the systematic natural language contrived for concocting, exemplification and exhibition of text using tags. It is a method to describe XML [1] articulation accurately. DTD's [2] may be called as entities in business. Mainly two types of declarations are made using document type definition i.e Internal and External. The Internal Structure section of the document itself is struck in within the DOCTYPE definition. In the External structure, bits to DTD [2] declarations are accommodated in external files because of security issues parsers. The Web browsers may be restricted from reviewing external fragments. DTD [2] narrates the arrangement of class of documents by means of element and attribute-list declarations. Element declaration labels the acceptable set of elements within the document, together with defining if and by virtue of what elements and runs of character details may be accommodated within every element. Attribute list declaration labels the acceptable set of attributes for every declared element, inclusive of the category of respective element value, or an exact set of authentic [4] values. DTD [2] mark-up declarations demonstrate which element type, attribute-lists, entities along with notations are granted in the arrangement of similar class of XML [1] documents. The second method of validation of extensible markup language documents is Schema [3]. Schema [3] defines the legal building blocks of XML [1] documents and is a written form of type of XML [1] document generally expressed using some composition of linguistic principles influencing the layout of elements on the structure and matter of that particular document of that kind. It conveys shared vocabularies and allows the compiler to compile principles build by user. XML offers the means of prescribing the design, matter and literals

[5] of the XML [1] document. It can be used by programmers to validate each piece of data content in a document. They can validate if it follows to the description of the element if it placed in.

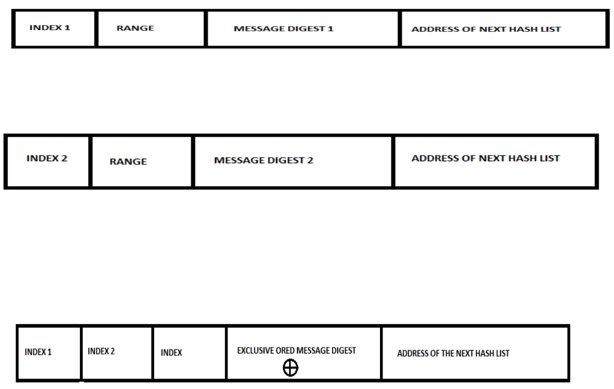
Message Digest
The process of creation of digital signature [6] has two major functions, one is signing and the other is sealing with a key. Signing is otherwise a mathematical summary or a hash code or simply a Message Digest (MD) [7]. It is a hexadecimal number that is mainly created by hashing formulas. If a single bit of the original message changes then there is a dramatic change in the message digest [7]. The most common cryptographic [8] hash functions that are mainly used are Message Digest- 5 (MD-5) [7], SHA-1(Secure hash algorithm-0), SHA-256[9], SHA-384, SHA-512 [10] etc. The MD5 algorithm bears a message inserted of various lengths and generates a 128 bit-hash code. It mainly involves the following steps: 1. Attach the lining bits 2. Adjoin the length 3. Initialize the MD buffer 4. Produce message in 512- bit block. The Secure Hash algorithm-1(SHA-1) is established on the rules of MD4 and MD5. The computation inputs a message of any length up to 264 bits and solves a 160-bit hash value. The algorithm involves the corresponding steps:1 Adjoin the lining bits i.e. message is padded with 1s and the 0s are appended 2. Append the length 3. Preparation of the processing functions and the processing constants 5. Initialization of the SHA [9] buffers 6. Processing the message in 512-bit blocks. The SHA-256 [9] dependent on MD4, MD5, and SHA-1. It executes a 512bit message chunk and a 256bit intermediate hash unit. The message that is to be hashed is lined with its extent in such a manner that the output is as many times as that of 512 bits long and after then it is processed into 512-bit message chunks. The message chunks are then parsed single at a time. The SHA-512[10] is a modified form as that of the SHA-256[9] that executes on eight 64-bit expression. The message that is to be hashed is lined with its extent in a manner that the output is as many times as that of 1024 and after then it is processed into 1024-bit message chunks. The SHA-384 is described in the same manner as SHA-512[10] but it is having mainly two exceptions, firstly the initial hash values are based on the fragmental chunks of the under root of the ninth through sixteenth primes and secondly the384-bit hash is acquired by docking the SHA-512[10]-based hash result to its left-most 384 bits.

Digital Signature-A Digital Signature [6] is a mathematical procedure or technique which is used for validating the credibility and integrity of the document as given in Fig 1. The main difference between a digital and handwritten signature is, that the digital signature [6] of a message is confidentially connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always looks the same. There are mainly two methods of encrypting the document, symmetric and asymmetric encryption. The Symmetric encryption is also named as the conventional encryption or one key encryption. In this technique, the key by which the message in encrypted is kept as a secret. The sender and receiver must keep the key secure. Compromised Key can lead to many security attacks and data leakages.

Digital Signature Standards
Digital Signature-A Digital Signature [6] is a mathematical procedure or technique which is used for validating the credibility and integrity of the document as given in Fig 1. The main difference between a digital and handwritten signature is, that the digital signature [6] of a message is confidentially connected with the message, and for different messages is different, whereas the handwritten signature is adjoined to the message and always looks the same. There are mainly two methods of encrypting the document, symmetric and asymmetric encryption. The Symmetric encryption is also named as the conventional encryption or one key encryption. In this technique, the key by which the message in encrypted is kept as a secret. The sender and receiver must keep the key secure. Compromised Key can lead to many security attacks and data leakages.The Asymmetric encryption mainly uses two approaches for credibility, the RSA [11] and the DSS [11] as shown in Fig 2 and 3 respectively. In the RSA [11] approach the sender's private key is applied on the message for generating the signature. The verification of the signature is done by applying the synonymous public key to the message as well as the signature through the verification process, which provides either a valid or an invalid result. On the contrary the DSS [12] approach uses the digital signature function. In this the message digest [7] of the algorithm is made through the utility of the hash function. The message digest [7] is then used along with the DSA computation in order to make the digital signature that is send with the message. The checking of the digital signature involves the utilization of the same hash function.

Distributed Message Digest
Distributed message digest [7] is a novel concept to enhance the basic functionalities of the standard digest or hash algorithm. In general, a single hash function is used to assert the consistency of a data transfer in a wireless environment, which is much required because data loss is very evidential in any network communications. The proven standards such as MD-5, SHA, SHA-256[9] are used in the form of digital signatures [6] with the help of proper security algorithms. Here the XML [1] and JSON have some unique features of dealing the data. The data can be dealt as a document object and it has some predefined hierarchy, so it is very easy to process the piece wise data rather than as a single unit so the proposed method aims to create a distributed message digest [7] that deals XML [1] and JSON separately.Here, the entire XML [1] document will be categorized as small chunks to

generate different message digest [7] for each, the XML [1] file is not divided physically. The indexing is followed here to manage all digests created for a single file. In general, the main file is virtually divided into equal sized small chunks. The chunk size is decided based upon the network bandwidth, file size, Signal Noise Ratio(SNR), bandwidth availability and other factors. In hash functions are applied to generate different hash codes for all virtually divided chunks. The Fig 1.4 depicts the overview of distributed message digest [7], in first level the message digest [7] is created using different standards according to the security requirements but in second level onwards the hash values are just X-OR-ed and the same process is iteratively followed to combine all X-OR-ed values into a single final digest. The sender will send the payload, message digests [7], X-OR ed values together. As depicted in Figure 5. sender decides to generate multiple digests for a given message. Here JSON and XML [1], have the basic properties of representing a document as an object model. The key elements are sub root elements can be used to divide the file virtually. After dividing the file virtually, multiple set of digests would be created based on the user's requirements. Usually, 2^N digests are created. In the second step, the message digests [7] are combined with the help of XOR functions at different levels.In first level, N/2 XOR list will be created to store the XOR-ed value of digest 1 and digest 2. The same procedure will be followed till it is getting converged to one XOR value.The traversals are bidirectional, that means the traversal can be made from digest to the converging point and vice versa, so that all digests and XORed values are interlinked in a hierarchical way. A sender will send message along with all digests and XORed values.A receiver generates multiple digests of the received messages. The key elements i.e. the sub root elements will divide the document virtually. After dividing the files, the message digests [7] will be generated for the small chunks of files that are created based on the user requirements. At the second level the message digests [7] are again combined with the help of XOR functions at different levels. In the first step, N/2 XORlist will be again created in order to store the XORed value of digest 1 and digest 2. The same procedure will be followed until we get one XORed value of digest 1 and digest 2. Now the decryption of the message digests [7] present at the sender's side will take place on the receiver's side. The receiver will now decrypt the message digests [7] at the sender's side. The decrypted message digest [7] of the sender is then compared with the message digest [7] generated at the receiver's side. Similarly, the digests at the second level are then compared with each other. If the digests at each level on the sender and receiver's side are same, it means that the data is not corrupted anywhere. The XORed values are checked at each stage and binary search approach is followed to locate the errors. This approach not only makes the error detection and correction accurate, but it also reduces the time for error location. Moreover, the user does not need to resend the file again.


INDEX 1
RANGE
MESSAGE DIGEST 1
ADDRESS OF NEXT HASH LIST
INDEX 2
RANGE
MESSAGE DIGEST 2
ADDRESS OF NEXT HASH LIST
INDEX 1
INDEX 2
INDEX
EXCLUSIVE ORED MESSAGE DIGEST
⊕
ADDRESS OF THE NEXT HASH LIST

II. PROPOSED ALGORITHM

The existing standards of the digital signatures [6] make use of the RSA [11] and the DSS [11] approach for asymmetric encryption. In this paper. It has been proposed a new method of distributed message digest [7]. Multiple hashes for a single document using various computations like MD5, SHA-1, SHA-256[9], SHA-512[10] etc. were generated. The message which is to be signed [6] is put to the hash functions and multiple secure hash codes of various lengths are generated. Exclusive or operation is then performed on these functions and a single hash code is generated. This hash code is now encrypted using sender's symmetric key in order to generate the signature [6]. Message along with the signature [6] are transmitted. At the receiving end the hash codes of the incoming messages are again generated. The hash codes along with the signatures [6] are then made the inputs into the verification functions. The outcome of the verification function is the value which is equal to the signature [6] component if the signature [6] is valid. If the outcome of the of the verification function at the receiver's end matches with the hash code of the sender, then the message is not corrupted. If there is a mismatch, then there will be some error in the message. Now in order to find the error, the sender will not have to send the whole document again. The multiple hashes that were generated on the recipient's end will be matched with those on the sender's end. The hash code in which there is a mismatch will have the error. Now only that part of the file will be transferred to the recipient again. Although this approach might take some time in generating various hashes, but the error detection will be much more efficient and the sender no longer needs to send the whole file again. Only that part of the file will be sent which is having the error. Thus, this approach is much more efficient in error detection and correction and the time taken in retransmission of the file is very much reduced.

Algorithm –

This algorithm explains about the generation of the message digest [7]; the input N takes the number of chunks that need to be digested.

```
STEP 1: Generation of Message Digest [7]
Input: Text File - M
Output: $H_{array}$
Declare N as number of Chunks
Declare FileSize=getSize(M)
Declare $C_{size}$=FileSize/N
start=0
End=$C_{size}$
Initialize I= ZERO
For I belongs to N
        $H_{array}$[I]=Hash (M, start, end)
        I: =I+1
        start=end+1
        end=end+csize
        If(end>=FileSize) then break;

End for
```

This algorithm illustrates the input text file M and the number of parts N in which we need to divide the file. The chunk size is then calculated by dividing the total file size with the number of parts into which the file is divided. Two variables are then declared, say start and end. Message Digests [7] are then generated for multiple chunks. The multiple message digests [7] that are generated are then stored in an arrays.

STEP2: Computing the Message Digest [7] of H_{Array}

The following algorithm gives the method to create multiple digests and their hierarchical ordering.

```
Input: $H_{Array}$ Elements
Output: XoRArray - n linked list
Declare N as the no. Of chunks
Initialize I=0, ans=0,j=0,k=0
//Following steps are repeated in the while loop//

While I<N
Start=I;
//create Structure () function is called//
MDlist Md1=createStructure();
MDlist Md2=createStructure();
//Hash array is stored in Md1 Digest//
Md1->dgst=$H_{Array}$[1]
//Next Hash Array value is stored in Md2//
Md2->dgst=$H_{Array}$[I+1]

//createXOR() function is called//
XORList XORL=createXOR();
XORL->XORValue=Md1->dgst ^ Md2->dgst;
XORL->FirtstIdx=Md1->idx;
XORL->SecondIdx=Md2->idx;
I=I+2
End While
```

createStructure():

Here basically the data structure MDList is defined with proper parameters,

Idx 1: Sequence number of message digest [7]

Range-It is the data range from where the message digest [7] is created.

Message Digest 1[7] -Actual message digest [7]

XORList: It is the next pointer to the XORList

The createStructure() function is called in relevant places to create a new data structure that can store the complete information about the digest.

```
Struct MDList
{
Idx Number
Range (Number1, Number2)
MDigest *dgst
Struct  XORList *Next;
}
allocateMemory(MDList);
return *MDList;
```

XOR List: This data structure consists of the following parameters

Idx1: Sequence number of the first message digest [7]

Idx2: Sequence number of the second message digest [7]

Index.: It consists of a floating point value.

XORDigest *XORValue: The computed message digest [7].

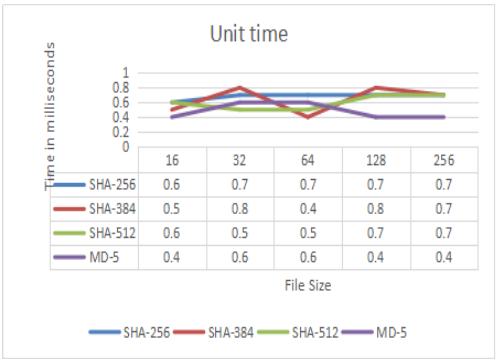
XORList *next: Consists of the address of the next pointer.

The XORList() function is called in significant places and it consists of the address of the next pointer. FirstIndex 1 and SecondIndex 2 also refer to the previous data structures also i.e. they perform the reverse traversal and refer to Message digest 1[7] and Message digest 2[7] in the previous data structures. XORDigest consists of the computed message digests [7]. XORList *next consists of the address of the next pointer.

```
Struct XORList
{
XORList *FirstIdx;
XORList *SecondIdx;
Index FLOAT;
XORDigest *XORValue;
XORList * next;
}
```

III. EXPERIMENT AND RESULT

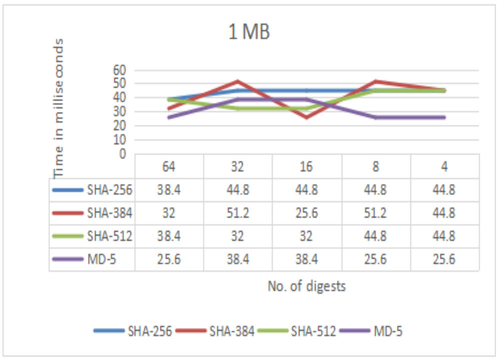
The JAVA code is written for implementing the proposed concept to test the time complexities in system configuration (Processor: Intel (R) core (TM) i5 – 3230 M CPU @ 2.60 GHz 2.60 GHz, installed memory (RAM): 8.00 GB (7.78 GB usable), System type: 64 – bit Operating System).



	16	32	64	128	256
SHA-256	0.6	0.7	0.7	0.7	0.7
SHA-384	0.5	0.8	0.4	0.8	0.7
SHA-512	0.6	0.5	0.5	0.7	0.7
MD-5	0.4	0.6	0.6	0.4	0.4

Fig.1 Unit Time Analysis

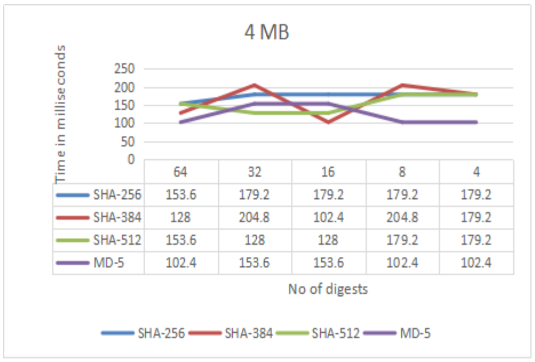
Fig.1 discusses about the unit time taken for creating digest of various sized files. The X-axis shows the file size in KB's and Y-axis gives the time in milliseconds. Different existing standards have been implemented to test the time complexities.



	64	32	16	8	4
SHA-256	38.4	44.8	44.8	44.8	44.8
SHA-384	32	51.2	25.6	51.2	44.8
SHA-512	38.4	32	32	44.8	44.8
MD-5	25.6	38.4	38.4	25.6	25.6

Fig.2 Time Analysis of 1 MB data

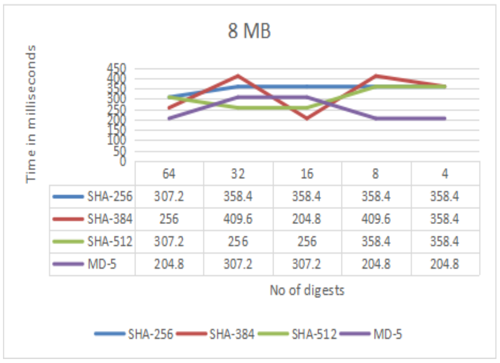
Fig.2 depicting about the time complexity of distributed digest. 1MB file is taken wherein 64 to 4 digest were created at the interval of half of the previous one. All existing standards were implemented to test the time efficiency. It is evidential that system takes very realistic time to manage the highest number of digest i.e. 64.



	64	32	16	8	4
SHA-256	153.6	179.2	179.2	179.2	179.2
SHA-384	128	204.8	102.4	204.8	179.2
SHA-512	153.6	128	128	179.2	179.2
MD-5	102.4	153.6	153.6	102.4	102.4

Fig.3 Time Analysis of 4 MB data

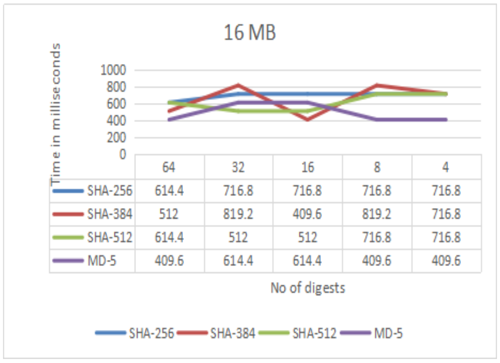
Fig. 3 is showing the same concept of distributed digest of 4MB data. These trials were made to test the initial hypothesis for extending the same concept to the bigger sized data. Here again, the upper bound of all existing standards is 250 milliseconds.



	64	32	16	8	4
SHA-256	307.2	358.4	358.4	358.4	358.4
SHA-384	256	409.6	204.8	409.6	358.4
SHA-512	307.2	256	256	358.4	358.4
MD-5	204.8	307.2	307.2	204.8	204.8

Fig.4 Time Analysis of 8 MB data

Fig.4 is showing the same concept of distributed digest of 8MB data. These trials were made to test the initial hypothesis for extending the same concept to the bigger sized data. Here again, the upper bound of all existing standards is 450 milliseconds.



	64	32	16	8	4
SHA-256	614.4	716.8	716.8	716.8	716.8
SHA-384	512	819.2	409.6	819.2	716.8
SHA-512	614.4	512	512	716.8	716.8
MD-5	409.6	614.4	614.4	409.6	409.6

Fig.5 Time Analysis of 16 MB data

Figure 9 is showing the statistical analysis of file sized 16MB. Here even if we create 64 digests where each digest size is 256 bytes still the time is realistic and negligible to check the consistency of a file unit-wise. When the file size is getting increased, the number of digest can be reduced to 32 for the realistic time complexities. When high end systems are used the time complexity still could be decreased. So all figures are showing the practical implementation and possibilities of doing the distributed message digest [7].

There is a new avenue for the researchers to optimize the problem further and to discover new methods of distributed digests.

IV. CONCLUSION

In general, the message digest [7] is used as a digital signature [6] for ensuring the data consistency and integrity, moreover, the authentication [4] is achieved through message digests [7]. The standard algorithms such as MD5, SHA are being used in all types of wired and wireless communications for implementing the digital signatures [6]. But some documents need special treatment since they are hierarchical and object oriented, some examples are XML [1] and JSON. This paper proposes a method of creating a distributed message digest[7] with a multi signature [6] method of signing a document .The main reason behind this work is to minimize the number of re-transmissions by detecting the accurate portions of corrupted data, wherein, the traditional methods will retransmit the entire data during data pollutions .This approach is more suitable for the object based documents since it can locate the error precisely and payload transmission will happen only for the corrupted portion of data. The data formats of XML [1] and JSON can support well this method and can leverage the full benefits of the proposed method. The main advantage of this method is the reduction of bandwidth usage and utilization of resources judicisiouly. In the era of web services, the documents are not the single units and only small sized data are getting transferred. So distributed message digests [7] are very essential to locate and correct the errors accurately. This paper opens a new gateway to the researchers for developing this concept further.

V. REFERENCE

[1] E. Bertino and E.Ferrari(2001) "XML and Data Integration" ,IEEE Internet Computing, Vol No.-5,pp 75-76.

[2] Dongwon Lee and Wesley W. Chu(2000), "Constraints-Preserving Transformation from XML Document Type Definition to Relational Schema", International Conference on Conceptual Modeling, Springer Vol 1920, pp 323-328

[3] Nicholas Routledge,Linda Bird,Andrew Goodchild(2002),"UML and XML Schema",IEEE Computer Society,Vol. 24,pp 157-166.

[4] A Khadijah Opatokun, Mohammed S. Nordin(2014),"Authentic Leadership Structural Validity: Higher Education Staff Notion on Authentic Leadership", International Journal of Research in Business and Technology, Vol 4, No 2,pp 127-140.

[5] Viji Rajendran,S. Swamynathan(2014),"Multi-threaded Priority based Semantic Crawler for Cloud Services", International Conference on Intelligent Information Technologies.

[6] Alexander Uskov, Hayk Avagyan(2014),"Fusion of Secure IPsec-Based Virtual Private Network, Mobile Computing and Rich Multimedia Technology", Intelligent Systems Reference Library,Springer, Cham,vol 84, pp 37-71.

[7] Alma Hossain, Md. Kamrul Islam, Subrata Kumar Das and Md. Asif Nashiry(2012), "Cryptanalyzing of Message Digest Algorithms MD4 and MD5", International Journal on Cryptography and Information Security, Vol.2.

[8] Thomas Loruenser,Daniel Slamanig,Thomas Länger(2016),"PRISMACLOUD Tools: A Cryptographic Toolbox for Increasing Security in Cloud Services",Availability, Reliability and Security (ARES), 2016 11th International Conference on 2016,IEEE.

[9] N Neelima1, Lekharaju Sai Siddhartha2, Chavali Meghana3, Shaik Sameer4, Shaik Ashika5, Vemulamada Naga Chandramouli6(2017),"Security in MANETS using Cryptographic algorithms",International Research Journal of Engineering and Technology,pp 127-132.

[10] E Fujisaki, T Okamoto(2013), "Secure Integration of Asymmetric and Symmetric Encryption Schemes",Springer, vol.26,no.1,pp. 80-101.

[11] B Huabai, B Shizhen(2012),"Two-Layer Fuzzy Comprehensive RSA-ANP-DSS Evaluation Model of Emergency Management Capacity about Enterprise Value Network", Elsevier,Vol.5,Pg93-98.