ISSN:0975-9646

Secure File Transfer by using Modular Encryption

D. Subhramanya Sharma[1], D.Geetha jyothi[2]

Dept. of CSE, Sri Sivani College Of Engineering,
Srikakulam, A.P, India

***Abstract*— Communication is one of the basic necessities of human beings. File transfer is one of the basic forms of communication. Reliability is the key issue raised due to complex nature of network and growth of computer science. In this paper we have devised a technique for file transfer which identifies whether some portion of the file is received corrupt or not, and if yes then exactly what portion is corrupt. This technique provides reliability by eliminating the corruption from a file, hence requiring less bandwidth of the network by reducing the amount of data to be re-sent in case of corruption. The reliability is ensured with the help of file signature generation method which we have devised in this paper. The beauty of this technique that it generates hashes which are not easy to break, hence ensuring security of the file. We have used TCP as the underlying protocol, whereas TCP is already considered to be reliable, but the fact is that it does not ensure the reliable transfer over the network due to the fact that it uses CRC which is still vulnerable to network conditions and malicious attacks. Our technique operates at the application layer and tries to finish the cope up with the reliability over the file transfer. We have also developed a prototype to test the integrity of our technique. Empirical results ensure the reliability of our technique. The emphasis of this paper is to provide users with the corrupt free file transfer over the network, so that their time and valuable resources might be saved**

I. INTRODUCTION

One of the basic necessities of living things in this universe is the communication. Information sharing is being done in many different forms and mediums like, signals, voice, and text etc. File sharing one of the basic forms of sharing, is being done since the human beings have started maintaining records. Files being transferred over the transmission medium contain valuable information. These files can be variable in size and importance to its users. One of the concerns of users transmitting files is that the file should reach accurately to the destination. That is, the file should reach complete, and uncorrupted.

File transfer protocol FTP commonly used protocol for transferring file over TCP allows platform independent data exchange [1, 2]. Size of information is getting increased with the growth in the field of computer science. Networks are growing rapidly and rushed with traffic by every passing moment. So, on one hand the information being sent to the network may not either always reaches to its destination or maybe corrupted during transmission. On the other hand the reliability of networks is never assured. Suppose that we are sending a file to some destination, and that we are sending a file to some destination, and during the transmission of file network link goes down. Now if the file size is too small it can be resend easily, but what if you were sending a large file size (giga bits) and almost at very last moments of transfer the file got corrupted. The situation becomes much worse if you have to resend the whole file. Similar kind of situation arises if you are sending/receiving important data and during transmission any kind of damage occurs to the file due to any reasons, the purpose of sending information dies out, and perhaps we need to retransfer the whole file again from the scratch.

In this paper we will present an algorithm that is specifically designed to overcome the drawbacks mentioned above. We call our technique as *Secure File Transfer by using Modular Encryption***.** The purpose of our technique is to provide reliability in transfer, and over come the problems which normally arise due to the underlying network. We have specially developed using file signature method this technique for the users transferring large file size. Other users are also the targeted users for this technique. Our technique is capable of identifying exact location of error at receiving site. This capability enables our technique to ask for the retransmission of only that part of application which is received corrupted and embed in the file at its position. This technique helps us saving a lot of time and valuable resources.

Rest of this paper is organized in the following manner: Section II contains the survey of existing file transfer techniques, Section III contains the proposed technique, Section IV contains the empirical test results of proposed technique and Section V concludes the paper **.**

2. BACKGROUND AND RELATED WORK

There is a number of file transfer application exists in the market developed on the basis of different techniques. Many of these are being used for professional purposes. Each of these applications is created for slightly different purpose. For example, there is a class of file transfer application which are called as "Secure file Transfer Applications". The purpose of file transfer application is to send or receive file only between the authenticated users etc.

This section encompassed the existing file transfer technique and existing file signatures method. The focal point of this survey is to find the file transfer technique which provides reliable transfer.

FU [3] presented a technique for file replicated on different sites. The authors are of that instead of transmitting whole file from one server to the other to check integrity of the files, the can use file signatures to accomplish the same task. This paper talks about creating the signature against whole file. But the technique they have proposed lacks the identification of exact location of error in the large file. The signatures method is only capable of identifying that the two file are not exactly same, that is one is changed.

Chang [4] presented a technique for multimedia databases. The authors have used signature for the sake of identification of the image icons of the database. Signatures used for this purpose lack in the sense that these signatures

are capable of telling that the icons are not same but these signature may not describe the exact location where two icons differ from each other. That is, the information provided by the signature is incomplete in sense that they tell that there is where the problem exists.

Chen [5] has presented signature method for multimedia object. The scheme uses hierarchy of object and signature are generated in the light of that hierarchy. This scheme may cause reduction in the disk access but to this part of hierarchy. This technique remains quiet. One of the reasons of this fact is that the signature used for this technique is not capable enough to identify the exact location the exact location of mismatch they are only made match perfectly.

Du [6] represents the extendable signatures. This is technique was introduced in the early days of file signature paradigm. The technique is providing only the basic functionality of hashing and signature files.

Li [7] presented a variation of signature files. The focus of this research activity is just towards the reposes time improvement of signature. This technique suffer from similar kind of weaknesses as mentioned in other techniques.

Lee [8] presents multilevel efficient signature which he represents as a much faster technique for text retrieval. If we critically review the technique discussed in this paper according to our interest, we come to the same conclusion that this signature method also lack the same properties as mentioned above.

Madduri [9] presents reliable file transfer for grid the reliability is here means that if some kind of error occurs during the file transfer, it can be started from the same point rather than starting from the beginning this paper does not include corruption free reception as reliability, that is the definition of reliability is incomplete in this is the on of this reasons if does not satisfy our requirements.

He [10] has proposed a UDP based data transfer application. The purpose of this file transfer application is bit different from the other application this application is designed for data transfer for high speed networks. So this application is mush more different from our domain. This technique is just an example of different purposed technique for file transfer.

The survey depicts that there are lot many different techniques exist for reliable file transfer over the network and many of those use signature file mechanism for the sake of providing reliability. There is one commonly seen problem in most of the techniques, that is, the definition of reliability is sort of incomplete. Due to this incompleteness of the definition, the techniques also remain silent on the portions which are not considered as a part of definition. There are different perspectives for all the existing techniques and their area of application may differ a lot but the underlying purpose of transferring file is the same which keeps them in single category. We have tried to identify and overcome the problems found in the existing techniques. One of the biggest problem we see is that the techniques are not either emphasizing on reliable transfer or techniques do not fulfill the requirements of reliability. The technique proposed by us mentioned in the next section tries to cope up with the above mentioned problems

3. PROPOSED RELIABLE FILE TRANSFER TECHNIQUE

We propose a technique with the main aim of securing the data as prevention is better than cure. Our main task is to send the file in a secure manner by following a three step process and detect the exact location where the corruption has occurred at receiving end. In order to accomplish this task we have devised an process in which

firstly on the Sender end the complete file gets encrypted using the RSA(Rivest Shamir Adolmen) Algorithm and later we split the complete encrypted file in terms of 1megabyte(can be varied) and later we apply Hash function by using the SHA-256 function class which is predefined in Java. Similarly on the Receiver end the file's Hash code will be compared and if the comparison is successful the file is later processed for joining and later on the Decryption process is followed. So as the file is undergoing through a three step process , the hacking of the file is highly impossible.

We propose a technique with the propose of handling corruption in file transfer. Our main task is to deleted the exact location where the corruption in file has occurred at receiving end . if you know the exact location of corruption .we will be able to ask the sending site and resend only the part of file received as corrupted as corruption

Algorithm: generated file with signatures

Input: user files in ASCII (fo)

Output: user file with signature appended at end of (fn)

Method : in order to apply hash function on each n bytes block of file we perform the following steps to make (in mod n) – 0of fo`

M ← calculated –length-of(fo)

N←length-of-Block

(Anyone128/256/512/1024/2048/4096/8192 bytes)

Res ←reserved 16 bytes

P← m mod n

Q← n-(p+res)

If(q>0)

F1> append Q zeros at the end of fo

Else if (q<0)

R← n%q

F1 ← append r zero at the end of fo

F1← append Res at the end of fo

In order to generate singnatures of F1,performe the following steps

L← calculate- length- of (F1)

Count ← l/n

For j ←l to count

S ← 0

S ←reverse $\sum_{A=1}^{n}((A+B)\forall(A\wedge B))$

Where B ← to_integer (to_char(A))

Sigh ← sig +to_binary(S)

F_n ← F_1 + sigh

In order to accomplish this task we have devised an algorithm which uses files signature method to identify the exact location of error. We call this file transfer technique as corrupt free file transfer using ftp

The above algorithm generates signatures against the data in file and appends those generated signature that this algorithm generates signatures for every block separately and

then those signatures are appended at the end of file as well this algorithm uses 16 bytes as reserved bytes . these bytes are used to send the original size of the file .block size in this algorithm (n) is dependent upon the preference of uses. We have performed testing of our technique represented later. In order to identify the best suitable block size for our technique

The method of identifying corruption at the receiving site uses the similar technique. Algorithm at receiving site first identifies the actual size of the file received. Then it separates the signatures from the received file. After doing this process file only contains the original data with appended zeros and 16 reserved bytes. The signatures are separated from the file this algorithm then again generates signatures of this original file and compares the signatures with received signatures. If signatures exactly match. If mean the file is received without errors. If match is not found. It means that the file is corrupted. now the question arise that what should be done now? Should we retransmit the whole file again? Or try to identify the portion of file which is corruption? If we consider it with the perspective of coast, as we have mentioned earlier that we largest user who are interested in larger file. Resending whole file from search will not be better option for lager file size. Even for file of small sizes. This is not a recommendable option to resend whole file. So we try to identify the problematic area in the file and try to ask sender application about only that part. Our file signature technique allows us to identify the corrupted area in the file and request the client to resend only that area. The mechanism of identifying the corruption area of file is illustrated below.

One very strong point about the proposed algorithm is that it fist divides the whole file into block of equal size. Signature for each block is separately generated and stored in the file. It means that the number of block in the file is exactly equal to the number of signatures generated. That is, each signature represents the unique block of data. The receiving site generates signatures of the file received after removing sending site signatures from the file. The signatures generated at sending site are then matched against the signatures generated against the receiving site matching of signatures of each block is done separately. If exact match is found. It means that the block is received accurately; otherwise the block is not received accurately. This technique makes us capable of identifying the exact block which are received corrupted. After the identification of corruption blocks.

Our receiving side asks sending side only for those block which are received corrupted.

The hashes generated by this algorithm are secure because on one hand. The algorithm generates the hashes of variable length so it would be almost impossible for the attackers to identify the start and the ends of single signatures and to change it. On other hand. Different block size in above hash function helps against the collision in the hash codes generated another reason to this fact is that each hash code generated on the basis of the ASCII character and its position in the block

4. EMPIRICAL EVALUATION OF TECHNIQUE

We have developed a prototypical application for our technique. The purpose of this prototype is to provide soundness to our idea of reliable file transfer we have critically evaluated our technique with two major perspectives, testing of identification of exact location of corruption in the file and what block size is suitable for transferring file by using our technique. The purpose of first evaluation is to ensure the reliability of our technique whereas the purpose of second evaluation is to utilize the client's resources efficiently and provide case for the people using this proposed technique

4.1 Integrity Testing Of Technique:

The signature generation and comparison provides assurance that either received file has not been altered or if it is attend then recognize exact location from where received file corrupt this test allows us to check that whether our technique identifies the exact location of error if error occurs or not. We have performed comparison of different signatures generation method and select the best suitable signatures generates method on the basis of result generates by different method

In the following scenario. We used a 22-byte ASCII values and generates signatures on it. Then we made little change in the input string and stored the signatures generates by the proto type

Following is the original string posed to the algorithm input and results generated (in ASCII and binary) by the algorithm are as under

Input:"This is a test program"

Result: sig=2422-100101110110

Following are the different scenarios representing variations in the original string

Scenario 1: changing t to s of substring "test"

Input: "this is a best program"

Result: sig=8322=10000010000010

Scenario 2: adding single space at the end string

Input: "this is a test program"

Result: sig=7922=1111011110010

Scenario 3: changing t to T of substring "test"

Input: "this is a test program"

Result: sig=122=1111010

Scenario 4: signature of empty string

Input: ""

Result: sig=82311=1010000011000011

The result of all above mentioned scenarios show that a slight change in the string causes major changes in the hashes generated by the algorithm this is the strength of our proposed algorithm that it is capable of detecting even a minor change in the file and it also capable of generating hash for an empty string. We have chosen this signatures method among many different option of signatures method. The reason of selection of this signature method is that is creates very different hashes for even small change in the block so it capable of deletion even a small change in any of the block of file.

4.2. Testing for the Right Size of Block:

We use block size as a unit of identification of error in our technique. This unit can vary according to the desires of the users of this technique. Block size can be selected as large or small, but there are problems with both of these lengths. We have to identify the best suitable size of block. In order to find the best suitable block size we have to test our algorithm for different block sizes and the select the best suitable block sizes on the basis of results generated by our prototype.

We have performed testing of our technique in real time environment under the following variables:

Test Data:
File Type: Word Document
Size: 1.0 MB
Test Systems:
Server:
Personal computer: Pentium IV
RAM: 256 MB
Hard Disk: 60GB
Internet Environment: connected at average 40Kbps
Client:
Compaq Laptop: Pentium IIII
RAM: 128 MB
HardDisk: 4GB
Internet Environment: Connected at average 40Kbps

Table 1 shows the testing results of our technique with different block sizes. The testing results are generated using TCP. We have performed our test 6 times with each block size and then calculate the average time, in nanoseconds, for transfer of each block size. The reason of making six different attempts is that the tests are performed in real environment and conditions of network may differ in different times. We have considered this factor in testing as well, so the tests have been performed in different timings.

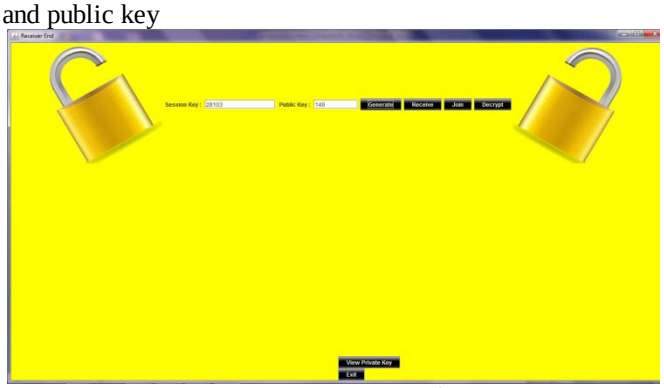
TABLE 1: Results of testing over TCP

	BS	128	256	512	1024	2048	4096	8192
	AS	1MB	1MB	1MB	1MB	1MB	1MB	1MB
	MS	1.04MB	1.02MB	1.01MB	1.0MB	1.0MB	1.0MB	1.0MB
	ST	3864	3556	3864	3808	3703	3800	3624
1	C	224843	100775	53010	28766	14837	7605	6830
	TM	3502	31586	31288	31275	31273	31308	6650
	S	152170	67991	36221	18345	10672	5229	2428
	TT	409515	200352	120519	78386	56782	44142	15908
2	C	234760	110207	65539	28852	15919	7955	6436
	TM	33100	31698	31275	32155	33238	30218	7176
	S	145446	70155	35041	18340	10468	5335	1428
	TT	413306	21206	131855	79347	59625	43508	15040
3	C	221167	96977	60147	28561	15208	7540	6877
	TM	32482	31602	31300	31275	32145	32150	6332
	S	14645	66207	36718	20017	11051	5429	1435
	TT	400103	194786	128165	79853	58404	45119	14644
4	C	222013	99857	54159	30006	16122	7655	6201
	TM	32490	31634	31295	31078	31070	31029	10483
	S	14821	68345	36531	19437	9879	5247	1434
	TT	4022720	1989836	121985	80521	57071	43931	18118
5	C	221850	101364	59168	29407	15208	7571	3931
	TM	32519	31578	31298	31300	32200	31200	31276
	S	146654	66561	36643	19330	12001	5319	3050
	TT	401023	199503	127109	80037	59409	44090	38257
6	C	225813	120650	56248	27991	14591	7445	9225
	TM	32611	31672	31200	32005	32100	31007	23569
	S	151011	67013	35432	18345	10837	5315	2530
	TT	409435	219335	122880	78341	57528	43767	35324
	AT	406017	20431	125419	79414	58137	44093	22882

Index Terms: BS(Block Size), AS(Actual File Size), MS(ModifiedFile Size) ST(Simple Transfer Time), C(Client), TM(Transmission Medium), S(Server), TT(Total Time), AT(Average Time).

5 OUTPUT SCREENS

Generation of keys: In this we are generating the session key and public key



Screen 1 Generation of keys

Choosing file:

In this screen we are choosing file which we are encrypting

Screen 2: choosing file

Choosing the Transfer button:

After the encrypting splitting hashing, we are transferring the file from senders side to receiving side

Screen 3: Choosing transfer button

At the receiver side after receiving joining decrypting we check your file named “orginal”

Screen 4

CONCLUSION

Reliability is one of the essential constructs of file transfer over network. In this paper we proposed a technique for reliable file transfer and proven the integrity of our scheme with the help of result generated by our prototype. we have provided case to the user of our technique by providing them variable block size for transfer. User are facilitated by the empirical result of our prototype to identify the best suitable block size for their situation. Reliability is provided with the help of file signature method devised in this paper. The uniqueness of this method is that it is capable of identifying the exact place of error in the file. This helps us saving our resources and time. We consider to be reliable, but the fact is that it does not ensure reliability to great extent. The reason behind this fact is that it uses CRC which is not secure in all the case and may not something identify the corruption. We have tried to overcome the shortcoming of TCP reliability problem in our technique. The technique works on the application layer and tries to identify the exact place of error in the file. It is designed in a way that it uses minimum bandwidth to retrieve heal the corruption for example. If the file is received corrupt. It would only request for that part of file which is corrupt and capable to join it with the rest of the file.

REFERENCES

[1] C.boris,r. Claudia, and s.marie-luise,”semantic cache mechanism for heterogeneous web querying”, computer networks,31(11-16):1347-1360,1999.

[2] k. Kato, and t. masuda, “persistent caching: an implementation technique for complex objects with object identity”, IEEE Transaction, july, 1992.

[3] A.w.fu, and S.C. cham. “locating more corruptions in a replicated file,” srds, p. 168, 15th symposium on reliable distributed system (SRDS’96), 1996.

[4]. J.W. Chang, and J. Srivastava, "Spatial Match Retrieval Using Signature Files for Iconic Image Databases,"icmcs, p. 658, International Conference on Multimedia Computing and Systems (ICMCS'97), 1997.

[5]. Y. H. Chen, A. J.T. Chang, and C. Lee, "Object Signatures for Supporting Efficient Navigation in Object-oriented Databases," dexa, p. 502, 8th International Workshop on Database and Expert Systems Applications (DEXA '97), 1997.

[6]. D.H.-C. Du, S. Ghanta, K.J. Maly, and S.M. Sharrock, "An Efficient File Structure for Document Retrieval in the Automated Office Environment," IEEE Transactions on Knowledge and Data Engineering, vol. 01, no. 2, pp. 258-273, Jun., 1989.

[7]. Z. Lin, and C. Faloutsos, "Frame-Sliced Signature Files," IEEE Transactions on Knowledge and Data Engineering, vol. 04, no. 3, pp. 281-289, Jun., 1992

[8]. D. L. Lee, Y. M. Kim, and G. Patel, "Efficient Signature File Methods for Text Retrieval," IEEE Transactions on Knowledge and Data Engineering, vol. 07, no. 3, pp. 423-435, Jun., 1995.

[9]. R. K. Madduri, C. S. Hood, W. E. Allcock, "Reliable File Transfer in Grid Environments," lcn, p. 0737, 27thAnnual IEEE International Conference on Local Computer Networks (LCN'02), 2002.

[10].E. He, J. Leigh, 0. Yu, T. A. DeFanti, "Reliable Blast UDP: Predictable High Performance Bulk Data Conference on Cluster Computing (CLUSTER'02),(2002)