DESIGN OF A CRYPTOGRAPHIC TAMPER DETECTION SCHEME FOR NETWORK SECURITY

B. SRINIVAS RAO, S.D.V. PRASAD

Abstract: In the present research work an attempt has been made to design and implement a tamper detection scheme that provides an additional procedure which detects tampering, given two signatures, whether one of them was generated by the forger. In this system, emails and files are signed digitally. The scheme automatically computes a hash based on the exact content of the email message, and then encrypts the value with the sender's private key. The recipient of the email will use their tamper evidence software to compute the same calculation. The matching of the calculation with the hash value is a proof that the message has not been altered. It ensures the data integrity, confidentiality and authentication.

Keywords: Network security, Tampering, Hash Function, Encryption and Decryption

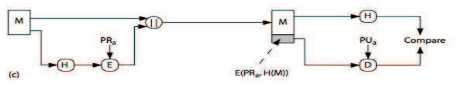
Introduction : Tampering is basically related to the data security which aims at ensuring that data is safe from corruption and that access to that is suitably controlled. Thus data security helps to improve privacy of the data like passwords, user info, etc. To avoid tampering of the data various encryption techniques are used like SHA1, MD5 etc., so that private data can't be reverse engineered to get the actual data. It is important for the Internet users to understand that the regular emails and file transfers offers no privacy and can actually be read by many people other than to whom it is sent to. The Internet Service Provider (ISP) probably keeps a copy on its computer and copies of documents sent from a networked computer are probably kept behind and all of the internet computers the email goes through on its way to the recipient can keep a copy. The administrators of all these computers can read the documents if they choose to and they can send it to anyone they might want to. Anyone that can intercept the document can alter the file content and anyone can send document that looks as if original sender sent it. Key exposure is a well-known threat for any cryptographic tool. For signatures, exposure of secret key compromises the corresponding public key. After the exposure is detected, the compromised keys can be revoked [1-6]. This detection of the exposure has previously been dealt with outside the scope of cryptography. Recently, Gene Itkis et al proposed a cryptographic scheme for tamper detection [7]. Considerable amount of research is being done in this direction. Many real-world applications wish to collect tamper evident logs for forensic purposes [8]. Matt Franklin presents a survey of key evolving cryptosystems in the public key setting, focusing on two main approaches: 'forward security' and 'intrusion resilience'. The essential feature of this design strategy is that the secret key changes over time, while the corresponding public key remains unchanged. Key evolving cryptosystems can limit the damage caused by an attacker who occasionally learns your secret key [9]. In our present work we propose a cryptographic solution to the above posed problem in the frame work of Itkis.

Tamper Detection Scheme : In the present research work an attempt has been made to design and implement a tamper detection scheme and its variants. The scheme automatically computes a hash based on the exact content of the email message, and then encrypts the value with the sender's private key. The recipient of the email will use their tamper evidence software to compute thesamecalculationasshowninFig.1. In Fig.1, at the Sender end the message M is subjected to a hash function H to compute a hash value H (M) which is encrypted by an encryption algorithm E using private key $PR_{a.}$ The cipher text E ($PRa_{,}$ H (M)) is appended to the plaintext message M and sent to the Receiver. At the receiving end, the same computation is performed on M using decryption algorithm and public key $PU_{a.}$ The computed value is compared with H (M) received by the Receiver. The matching of the calculation with the hash value is a proof that the message has not been altered.

Design of the Tamper Detection Scheme : The entire procedure consists of processes at Sender end and Receiver end. At Sender side the text is first encrypted. Hash code is created for the original text by using hash table techniques and DSA algorithm. The encrypted data and hash code are merged and sent through a data communication channel. At the Receiver side the received text is decrypted .A hash code is created for the received text and then compared with the hash code received. If both the hash codes do not match then the received text is tampered, thus providing evidence. The scheme consists of the following modules.

Encryption : Encryption is the conversion of data into a form, called a cipher text that cannot be easily understood by unauthorized people. Suppose Alice wants to send a message M to Bob. Alice creates the cipher text C by exponentiation C = I mod N, Where

E and N are Bob's public key. Alice sends C to Bob. The input is plain text file and the data is encrypted and output will be the encrypted file. The purpose of Encryption is to encrypt the message digest or intelligent plain text file.



14

Fig.1 Authentication with Public Key Encryption using a Hash function

Hash Function : A hash value is generated by a function H of the form H =H (M) Where M is a variable length message and H (M) is the fixed length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates the message by recomputing the hash value. Because the hash functions are typically quite complex, it is useful to examine next some very simple hash functions to get a feel for the issues involved. We then look at several approaches to hash function design. The purpose of a hash function is to produce a "fingerprint "of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties can be applied to a block of data of any size produces a fixed length out put H (X) is relatively easy to compute for any given x , making both hardware and software implementation practical. For any given code h, it is computationally infeasible to find x such that H(X) =h. This is some times referred to in the literature as the one way property for any given block x , it is computationally infeasible to find y!=x with H(y)= H(x).this is some times referred to s weak collision resistance. It is computationally infeasible to find any pare (x, y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.

RSA Algorithm : Ronald Rivest, Adi Shamir and Leonard Adleman developed the RSA system in 1977. RSA is a public-key cryptosystem. The algorithm is as follows:

Take two large prime numbers, P and Q.

Compute their product,

$$N = P*Q.$$

Compute the function if N as

$$F (N) = (P-1) (Q-1).$$

Choose a number, E less than N and relatively prime to F (N). This means that E and F (N) have no common factors other than 1.

Find another number D such that

D*E mod F (N) = 1.

The values E and D are called the public and private exponents respectively. The public key is the pair (N, E) and the private key is (N, D). Decryption by someone who doesn't know E would involve finding the Dth root of the encrypted message (mod N) which is accepted as a computationally intractable problem even with fairly small numbers it would take powerful computers hundreds of years to do this. It is also important to determine E given the public key. To find E, you Need to know D and M. As N = P*Q and F (N) = (P-1) (Q-1), to find M you would have to break N up into its prime factors. Again, this is a computationally intractable problem. Provided large prime numbers are used for P and Q. They should be on the order of 10^{75} to 10^{100}. Then, even if we take powerful computers it takes hundreds of years to determine the secret key from the public key.

Merging : Merging is a technique that brings together several existing process models and creates a new process model. Merge the encrypted message and the Hash value and send the message to the receiver by E-mail. Both the files i.e.; encrypted text and hash file are combined and merge.mer file will be generated. The merge.mer file is transferred to the receiver.

Decryption : Decryption is the process of converting encrypted data back into its original form the cipher text is converted back to the normal text. To decrypt the cipher text C, Bob also exponentiates: $M = C^D$ mod N. The relationship between E and D ensures that Bob correctly recovers M. Since only Bob knows D, only Bob can decrypt the message.

Demerging : In the process of demerging the data will be separated. The encrypted text and the SHA code are separated in the demerging process. The file merge.mer which is received from the sender is demerged in the receiver side. The received file merge.mer is separated to encrypted text and the hash code. The data is divided into encrypted text and hash code.

Tamper Detection : The input recovered hash code and created hash code are compared for the tamper detection. We compare the hash code which is sent from the sender and the hash code which is generated after the process of decryption. Comparing these two hash codes, we can display the message which describes intelligent message is tampered or not. The received file is authenticated if the two hash codes are same and no tampering is done. The file is not authenticated if manipulations are done in the hash code and send to the receiver. Minute manipulations can be detected in the tamper detection.

System Design : In logical database design we approach database development from two perspectives. First, we transform the conceptual data model into a standard notation called relations, based on relational database theory. During bottom-up analysis we verify exactly what data are to be maintained in the database and the nature of those data as needed for each transaction, report, and so forth. The final step in logical database design is to transform the combined and reconciled for well-structured data specification. Fig.2 and Fig.3 show the sequence diagram both in sender and receiver perspective.

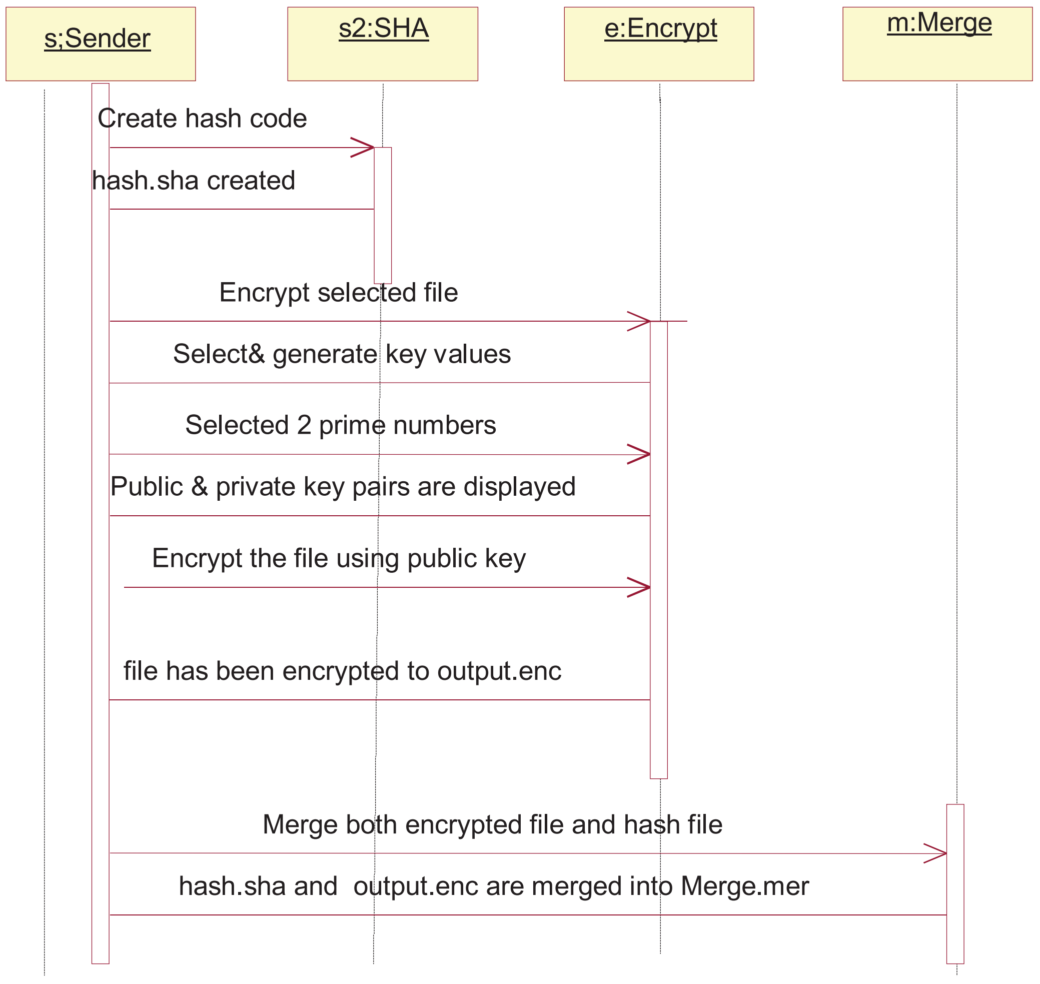

s;Sender
s2:SHA
e:Encrypt
m:Merge
Create hash code
hash.sha created
Encrypt selected file
Select& generate key values
Selected 2 prime numbers
Public & private key pairs are displayed
Encrypt the file using public key
file has been encrypted to output.enc
Merge both encrypted file and hash file
hash.sha and output.enc are merged into Merge.mer

Fig.2 Sequence diagram from sender' view

System Requirements :
Hardware Requirements
Intel Processor (Giga hertz)
RAM 128 MHz
Network Connection Card
Software Requirements
Operating System: Windows
Java 2 Runtime Environment
Java Standard Development Kit 1.4.0
Java Servlet Development Kit 2.0 or higher versions.

System Testing

Testing is done to ensure reliability of the software, to recover form errors and from errors and unknown bugs that are present. During testing, the program to be tested is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program delivers the performance as expected. There are chances for various errors to occur during any phase of the software development cycle. Verifications are done at the output of each phase. Each modules and sub modules are checked for errors at the output of each phase.

Testing Strategies : Test cases are devised with the purpose of finding errors. A test case is a set of data that the system will process as normal input. For this system, the test data is devised to check if the adjustments are done correctly. The other test cases devised is to check the situation in which no data is

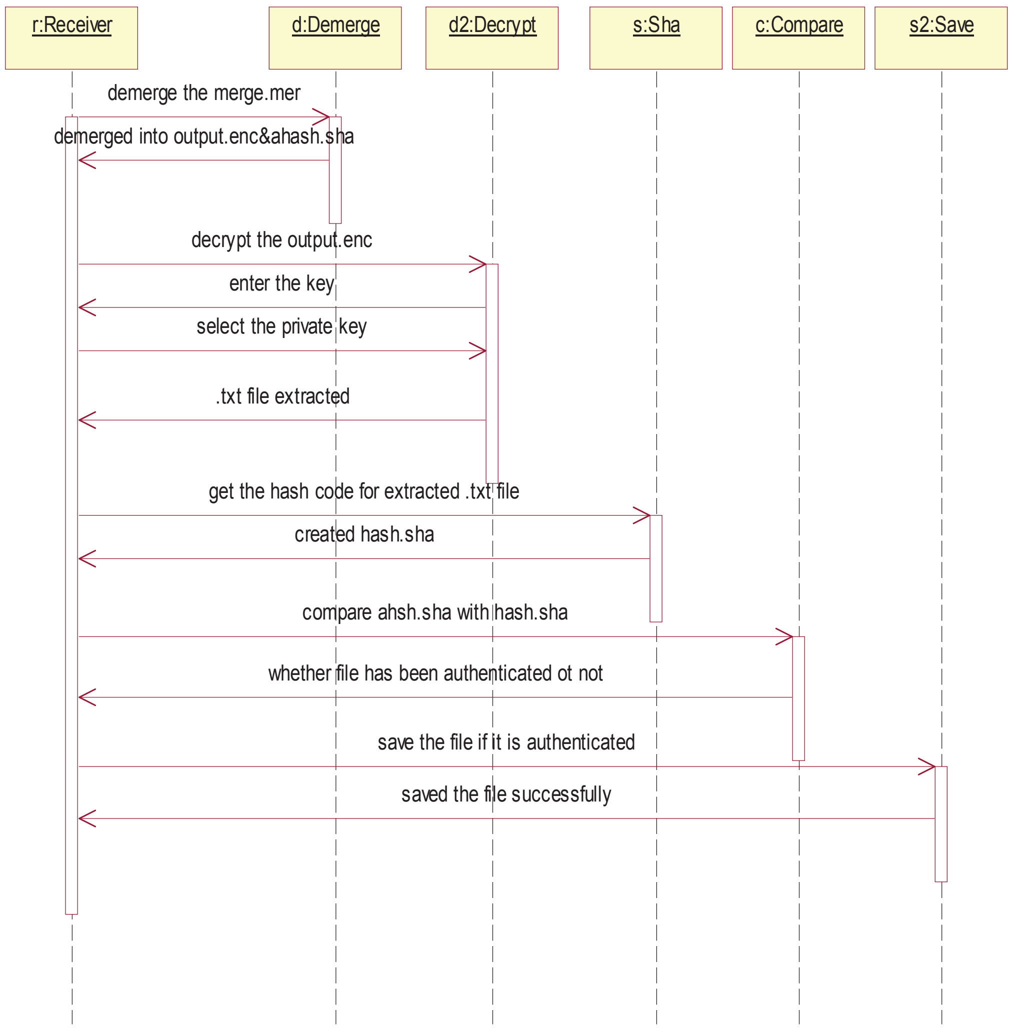

r:Receiver
d:Demerge
d2:Decrypt
s:Sha
c:Compare
s2:Save
demerge the merge.mer
demerged into output.enc&ahash.sha
decrypt the output.enc
enter the key
select the private key
.txt file extracted
get the hash code for extracted .txt file
created hash.sha
compare ahsh.sha with hash.sha
whether file has been authenticated ot not
save the file if it is authenticated
saved the file successfully

Fig.3 Sequence diagram from Receiver' view

available for adjustment for a specific condition. System testing is designated to uncover weakness that was not detected in the earlier tests. The total system is tested for recovery and fallback after various major failures to ensure that no data are lost. An acceptance test is done to ensure the user about the validity and reliability of the system. The philosophy behind the testing is to find error in project. There are many test cases designed with this mind .the flow of testing is as fallows: Code testing, Unit testing and System testing.

Code Testing : Specification testing is done to check if the program does what it should do and how it should behave under various condition or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing. This strategy examines the logic of the program. Here only syntax of the code is tested. In the testing syntax errors are corrected .To ensure that the code is perfect we performed unit testing and system testing.

Unit Testing : The first level of testing is called unit testing. Here different modules are tested against the specification produced during the design of the modules. Unit testing was done to test the working of the individual modules with test oracles. Unit testing comprises the set of tests preformed by an individual programmer prior to integration of the units into a large system. A program unit is usually small enough that the programmer who developed it can test it in great detail. Unit testing focuses first on the modules to locate errors. There errors are verified and corrected so that unit perfectly fits to the project.

System Testing : The next level of testing was system testing and acceptance testing. This testing was done to check if the system has met its requirements and to find the external behavior of the system. System testing involves two kinds of activities: Integration testing and Acceptance testing.

Integration Testing : The next level of testing is called the integration testing. In this many tested modules are combined into subsystems, which were then tested. Test case data was prepared to check the control flow of all the modules and to exhaust all the possible inputs to the program. Situation like treating the modules when there is no data entered in the file was also tested. This testing strategy dictates the order in which modules must be available, and exerts strong influence on the order in which the modules must be written, debugged and unit tested. In the testing all the modules / units on which unit testing is performed are integrated together and tested altogether.

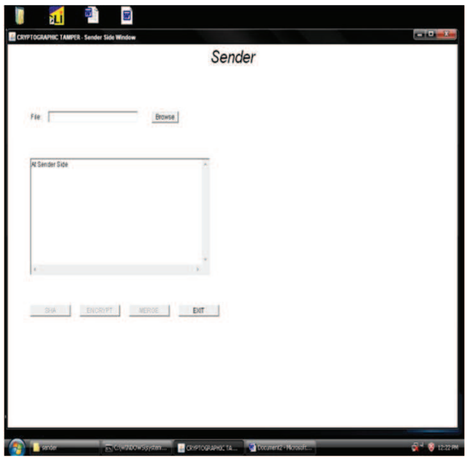
Acceptance Testing : This testing is performed finally by user to demonstrate that the implemented system satisfies its requirements. The users give various inputs to get required outputs.

Specification Testing : Specification testing is done to check if the program does what it should do and how it should behave under various condition or combination and submitted for processing in the system and it is checked if any overlaps occur during the processing.

Performance Time Testing : Performance time testing is done to determine how long it takes to accept and respond, the total time for processing when it has to handle quite a large number of records. It is essential to check the exception speed of the system that runs well with only a handful of test transactions might be slow when fully loaded. So testing is done by providing large number of data for processing.

Implementation : The implementation of the Tamper Detection Scheme is as shown in screen shots 1 to 14. Screen shots 1 to 7 depict the various actions that take place at the sender side. At sender' side the following actions take place: creation of hash code, encrypting the selected file, generation of public and private keys, encrypting the file using public key, and merging of encrypted and hash file. The merged file is sent to the receiver. At the receiver's end reverse process takes place. The merged file is demerged into encrypted file and hash file. The encrypted file is decrypted using private key. The hash code is computed to the decrypted file. The computed hash code is compared with hash code received from the sender. During the comparison if both hash codes are same it is supposed that no tampering has occurred. Otherwise it can be understood that tampering has occurred.

7.1 Senders' side:

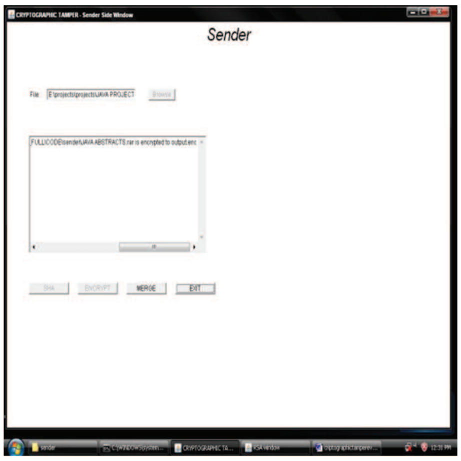

CRYPTOGRAPHIC TAMPER - Sender Side Window

Sender

File:

At Sender Side

Fig. 7 Screen Shot -1

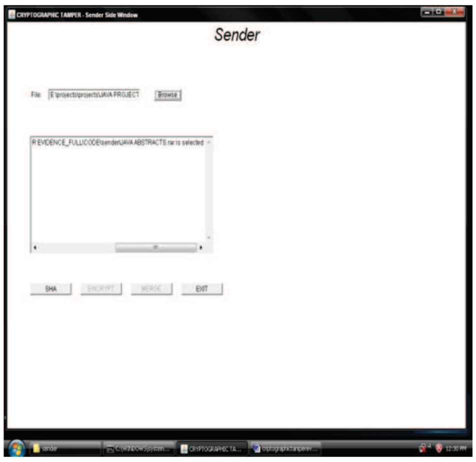

CRYPTOGRAPHIC TAMPER - Sender Side Window

Sender

File: E:\projects\projects\JAVA PROJECT

FULLCODE\sender\JAVA ABSTRACTS.rar is encrypted to output.enc

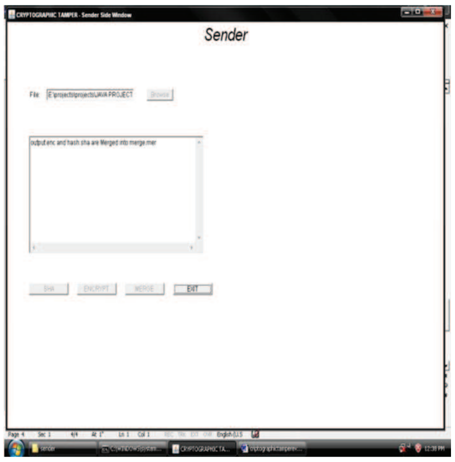
Fig. 9 Screen Shot -4


CRYPTOGRAPHIC TAMPER - Sender Side Window

Sender

File: E:\projects\projects\JAVA PROJECT

R EVIDENCE_FULLCODE\sender\JAVA ABSTRACTS.rar is selected


CRYPTOGRAPHIC TAMPER - Sender Side Window

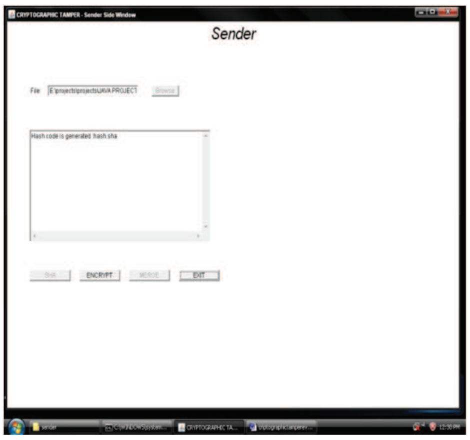
Sender

File: E:\projects\projects\JAVA PROJECT

output.enc and hash.sha are Merged into merge.mer

Fig. 8 Screen Shot -2

Fig. 10 Screen Shot -5


CRYPTOGRAPHIC TAMPER - Sender Side Window
Sender
File
Browse
Hash code is generated: hash.sha
SHA
ENCRYPT
MERGE
EXIT

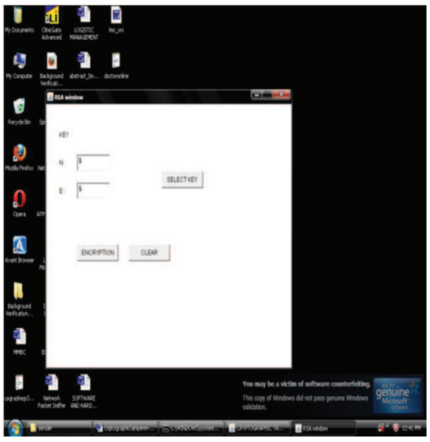
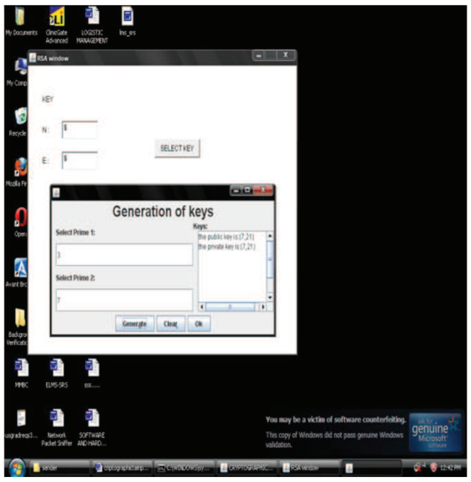


Fig. 9 Screen Shot -3

Fig. 11 Screen Shot -6



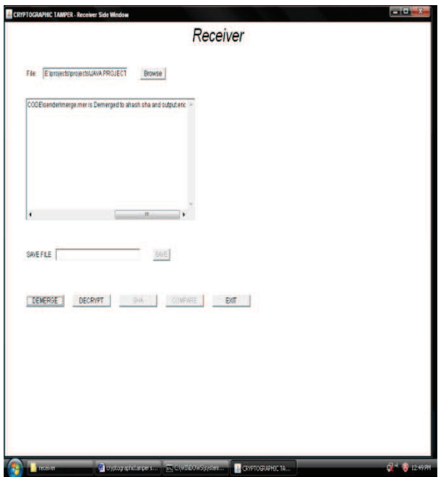
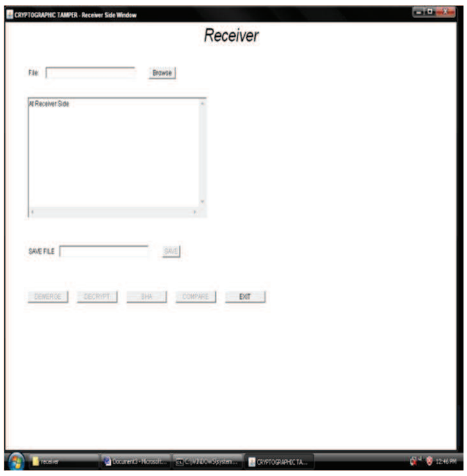


Fig. 12 Screen Shot -7

Fig. 15 Screen Shot -10

7.2 Receiver side



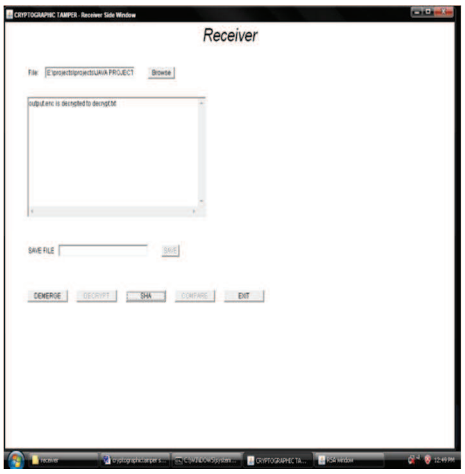
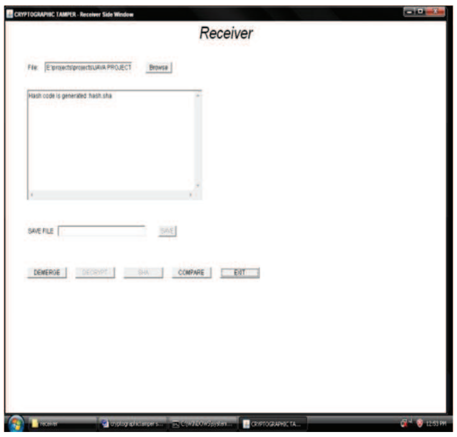

CRYPTOGRAPHIC TAMPER - Receiver Side Window
Receiver
File
Browse
output.enc is decrypted to decrypt.txt
SAVE FILE
SAVE
DEMERGE
DECRYPT
SHA
COMPARE
EXIT

Fig. 13 Screen Shot -8

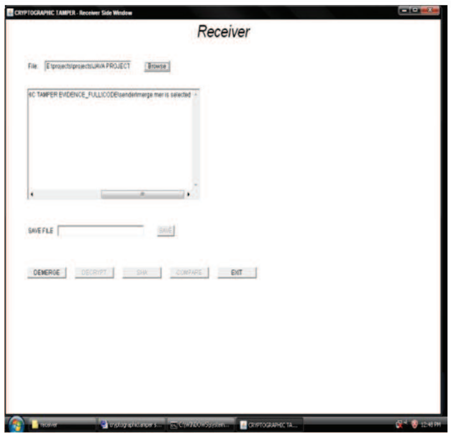
Fig. 16 Screen Shot -11



File: E:\projects\projects\JAVA PROJECT

Hash code is generated :hash.sha

Fig. 14 Screen Shot -9



File: E:\projects\projects\JAVA PROJECT

IC TAMPER EVIDENCE_FULL\CODE\sender\merge.mer is selected

Fig. 17 Screen Shot -12

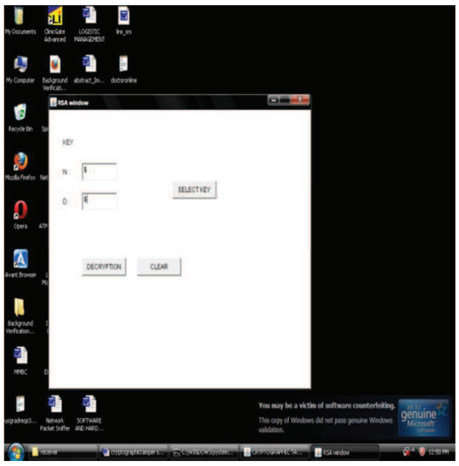
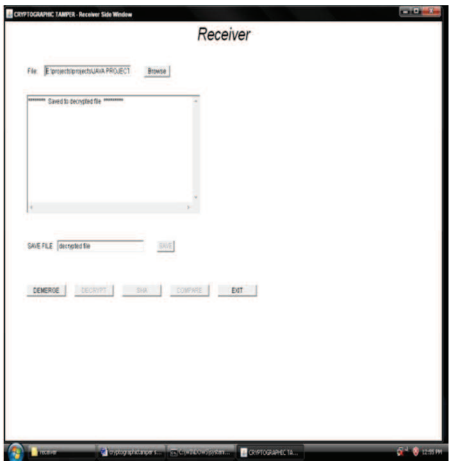


Fig. 18 Screen Shot -13



File: E:\projects\projects\JAVA PROJECT

******* Saved to decrypted file ********

SAVE FILE: decrypted file

Fig. 19 Screen Shot -14

Conclusion: The challenge of cryptography is developing a system in which it is impossible to determine the key. This is accomplished the use of a one-way function. With a one-way function, it is relatively easy to compute a result given some input values. To encrypt data, enter the data "plain text" and an encryption to the encryption portion of the algorithm. To decrypt the "cipher text" a proper decryption key is used at the decryption portion of the algorithm. The work done herewith has given a lot of insight into the working of the Networking-programming environment. The program written for encryption and decryption using IDEA Algorithm is tested on several textual files and results are observed. The results are in the form of screen shots for effective presentation. The program could achieve a better secure transferring of files between the server and various clients. The program written could be extended to higher order to achieve a better secure transferring of files between server and the various clients. In future different types of hash functions may be used for improvement of the present scheme.

References:

1. Gasser, M. Building a Secure Computer System, New York: Van Nostrand Reinhold, 1988
2. Gollmann, D. Computer Security, New York: Wiley 1999
3. Cheswick w and Bellovin, Internet Security: Repelling the Wily Hacker.
4. Felten, E. "Understanding Trusted Computing: Will Its Benefits Outweighs Its Drawbacks?" IEEE Security and Privacy May 2003.
5. "Improving Tamper-Evident Packaging: Problems, Tests and Solutions", Jack L. Rosette, 1992
6. "Tamper Evident Microprocessors", Adam Waksman and Simha Sethumadhavan, 2010
7. Gene Itkis, Cryptographic tamper evidence, Proceedings of the 10th ACM conference on

Computer and communications security, October 27-30, 2003, Washington D.C., USA

8. Scott A. Crosby and Dan S. WallachSSYM'09 Proceedings of the 18th conference on USENIX Security symposium ,USENIX, Association Berkeley, CA, USA ©2009.

9. Matt Franklin, International Journal of Security and Networks, Volume 1 Issue ½, Interscience Publishers, Geneva, SWITZERLAND, 46-53, 2006

***

Department of Computer Science and Engineering,
Malla Reddy Engineering College,
Maisammaguda, Dulapally, Hyderabad-500014,
bmsssaditya_1997@yahoo.co.in