Design of a Secure File transfer System Using Hybrid Encryption Techniques

Abdeldime M.S. Abdelgader, Lenan Wu, Mohamed Y. E. Simik and Asia Abdelmutalab

***Abstract*—Due to the recent innovations in the internet and the network applications and the wide spread of internet and networks, it is now completely possible to conduct electronic commerce on the internet or through the local area networks, and the wide spread of computer and communication network promoted many users to transfer files and sensitive information through the network, this sensitive data requires special deal. This work presents a security system that can provides privacy and integrity for exchanging sensitive information through the internet or the communication networks, based on the use of recently developed encryption algorithms, such as AES, IDEA and RSA. The aim of the work is to develop a simple file transfer system that can obtains privacy, integrity and authentication for the file transfer process. The proposed system uses symmetric cryptography system for securing file transfer while using public key cryptosystem and one way hash function to provide integrity, authentication and key distribution. The system is developed while putting into consideration the optimization of the communication channel and the speed of the encryption process.**

***Index Terms*—Network security, computer security, RSA, IDEA, File transfer**

I. INTRODUCTION:

For the first few decades of their existence, computer networks were imparity used by university researches for sending email, and by corporate employees for sharing printers and resources and managing data. Under these conditions, security did not get a lot of attention. But now, as millions of ordinary citizens are using networks for banking, shopping, E-commerce and filling their tax returns, network security is looming on the horizon as a potentially massive problem. Security is a broad topic and covers a multitude of sins. In its simplest form, security system concerns with making sure that nosy people (intruders) cannot read or worse yet, modify messages intended for other recipients [1-4]. It also concerns with people trying to access remote services that they are not authorized to use. The increasing awareness of security system and communications vulnerability has brought cryptography [4-6] out of its traditional shadow world and puts it into the forefront of modern computer technology.

Manuscript received May 30, 2015; revised, July 27, 2015.

Abdeldime M**.**S**.** Abdelgader is a lecturer with Karary University, Khartoum-Sudan, now he is a PhD candidate with Southeast University, School of Information Science & Engineering, Nanjing, 210096, China, corresponding author phone: +8613584003982; E-mail: abdeldime@hotmail.com.

Lenan Wu is a full professor with Southeast University, School of Information Science & Engineering, Nanjing, 210096, China, wuln@seu.edu.cn.

Mohamed Y**.** E**.** Simik is a PhD candidate with Harbin Engineering University, Information & Communication Engineering College. Harbin, China, mocimic@hotmail.com

Asia Abdelmutalab with Karary university computer and electrical departments, Email: asiatalab@gmail.com.

Since software and dedicated chips readily deal with the most complex mathematical calculations, simplicity of use is not an overriding concern. The most untutored user may now easily implement the most highly sophisticated package. Since cryptographic packages suggest something to hide the spy still has the problem of anonymity but at least for the rest of us, they are now a standard product. Unfortunately, for a variety of reasons, this is not quite the same thing as saying that electronic privacy is readily available. Security solutions has been presented in the literature to provide security services[7, 8]. These solutions mainly relies on cryptography and hash functions[9]. Cryptosystems are divided based on the used key into symmetric and asymmetric[10, 11]. Both of them have their own advantage and many limitations.

This paper presents a secure file transfer system using the advantage of both symmetric and asymmetric cryptosystems. The proposed system utilizes the advantages of the asymmetric cryptosystem to solve the limitation of the symmetric ones and vice versa. The proposed security system uses the advantages of symmetric cryptosystem, mainly International Data Encryption Standard (IDEA)[12-14], to provide fast and robust security, while utilizing the properties of asymmetric systems and hash functions for providing key distribution, integrity and authentication. This paper describes how IDEA, RSA [13, 15, 16] and the hash functions can fit in a network security system and file transfer. The design and the implementation are presented. The proposed scheme shown in Fig.1 can be used to securely send files and sensitive data from client to server or from server to client.

The rest of the paper is organized as follows. Section II provides a description about the proposed security system from both sender and receiver sides. In section III we provide a description to the implementation settings. Section IV tackles further details about the main components of the system. The result analysis is provided in section V, while section VI concludes this work.

II. THE PROPOSED SECURITY SYSTEM

A scheme has been proposed to provide a security for the computer network taking advantages of high security that can be provided by the IDEA, RSA and the hash functions. The proposed scheme shown in Fig .1 has two paths from the sender (client) to the receiver (server) and vice versa. The first path provides the security for plaintext and to the hash using the IDEA algorithm, whereas, the second path provides the security for the session key and integrity service using the RSA algorithm and hash function. It is worth mention that the network between the client and the server is assumed to be un-secure network (USN). The operation of this scheme can be explained by the following procedure to send a message from the client to the server. However, before starting the demonstration of the system design, we have to consider the notations described in Table 1.

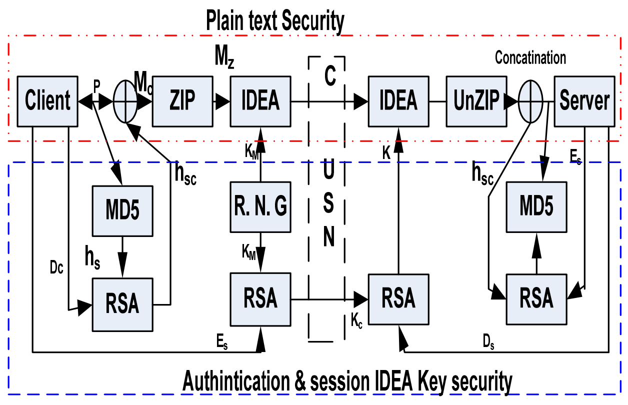


Fig. 1 The Scheme Model of the Proposed File Transfer Security system

A. *Sender Side*

Let the message that the client wants to send to the server is P. The scheme uses the plaintext P and the hash function Message Digest (MD5), which converts the client's plaintext file as a message digest $h_s = md5(P)$. This latter h_s is signed using the client's private key D_B with the RSA algorithm [16] to give

$$h_{cs} = {h_s}^{D_b} mod\ (n) \qquad (1)$$

Where, n is the product of two large prime numbers P_C and Q_c, which is generated by the RSA algorithm [15], [16].

The encrypted message h_{cs} is concatenated with the plaintext P to form M_C. Then, to provide more security and to reduce the time required for encryption, a compression algorithm has been applied to the message M_C to give the compressed message M_z.

TABLE 1. USED NOTATIONS AND DESCRIPTIONS

Notation	description
USN	Unsecure network
R.N.G	Random number generator
PGP	Parity good privacy
MD5	Massage Digest 5
P	Plain text
h_0	Hash function
K_M	The session key
h_c	Cipher hash
h_{cs}	Cipher sign hash
D_B	Private key of the client
D_s	Private key of the server
E_B	Public key of the client
E_s	Public key of the server
M_c	The concatenated version of the message and encrypted hash
M_z	Compress version of M_c
C	The ciphertexts
ZIP	Compress function

The compressed message M_z is then splinted into several groups each of 64 bits length, to be encrypted by the IDEA algorithm using the one time session key K_M of 128 bits length to produce the ciphertexts C. This K_M later is transmitted, after it has been encrypted, as a session key together with the client ID. The session key encryption is achieved by the most popular RSA algorithm using the server (receiver) public key E_s. The encrypted session key and the ciphertexts C are then concatenated to form a new file which will be sent by the client to the server using one of the transfer options.

B. *Receiver Side*

On the receipt of the packet from the client, the server (receiver) removes the client ID and recovers- decrypts - the session key K_M using RSA and the server private key D_s. The server then uses the session key and the symmetric encryption algorithm, which is IDEA in this paper, to recover the compressed version of the message M_z. It then uses the same compression algorithm to recover M_c which includes the original plaintext and its signed hash. For authentication and integrity purposes, it is then up to the server to check the client's plain file hash by using the same hash function and RSA algorithm but with client public key E_B. The scheme is designed to achieve a conversation between the client and the server. Therefore, the same procedure described is used when information is sent from the server to the client.

III. SOFTWARE IMPLEMENTATION OF THE SYSTEM

When started with experimental part for the system, the flowcharts of the procedures of the main component parts of the system are drawn. This will be described in detail later. These flowcharts are then converted to programs written in Borland DELPHI 6.0 language however, it is better to use JAVA language for programming this system because of its high accuracy and flexibility on computer security system design.

The system is efficiently operated on a computer (Pentium III) with the following specifications:

- Speed 833GHz full cache.
- 128 Mbytes RAM.
- 30 G bytes hard disk.
- LAN card 10/100 bits

When the programs were run in this computer, the average of response time was smaller than the response time mentioned in the theory, depending upon the certain size of the files that are encrypted.

A. *Computer Simulation of Client Side:*

When the system is installed to any computer, the followings information need to be provided.

1. Client name (ID).
2. Client Password.
4. Client e-mail address [option].

After the client answered these questions, the system will generate two keys (public and private) from RSA program and give the client a list of the other clients that are already installed in the system. The private keys of the client are saved in client's private database to keep this key physically more privacy and secrecy. The public keys are saved into a simple database file and are known to all users. The system components and their results are explained in the following sections.

IV. MAIN COMPONENTS OF THE SYSTEM

A. *Hash function procedure:*

This hash function program includes three options as shown in Fig.2:

(1) The first one used to hash the plaintext. The resulted hashed data can be used either for authentication or integrity purposes, as shown in Fig 2.(b)

(2) The second one can be used for comparing a received hash with another one generated at the recipient side to insure that the plaintext has not changed.

The hash program (Md5) performs the first task by reading the plaintext file as a binary file, and use the hash function algorithm to generate the 128-bit hash of the file, and the second task by reading the received hash and compare it with the local generated one, bit by bit and verify whether it is the same or not.

B. RSA procedure

The encryption key is a pair of numbers (e, n) where n is a product of two prime numbers p and q and e is a number relatively prime to the number $\emptyset(n) = (p-1) \times (q-1)$. The decryption key is a number d such that $d \times e = 1 mod((p-1) \times (q-1))$. The plaintext is broken into blocks of size b, treating each block in as a binary number (each number is less than n). The corresponding ciphertexts block is obtained by computing $c = m^e mod\, n$. In the decryption the cipher text is also broken into blocks of size b (by the way encryption is done, each of these blocks will represent a number less than n). For a block c_1, the corresponding plaintext m_1 is obtained by computing $m_1 = c_1{}^e mod\, n$.

Note that the key size here is not fixed unlike the secret key systems. The receiver generates his key using the following steps:

(1) Fix the size of n (it is usually 512, 1024 or 2048 bits).

(2) Generate two random prime number p and q of roughly equal size so that their product n has the size as fixed above (if n has 512 bits, then p and q should each be about 256 bits long).

(3) Generate a random number e of size close to that of n such that e and number $\emptyset(n) = (p-1) \times (q-1)$ have no common factors.

(4) Compute d such that $d \times e = 1\, mod((p-1) \times (q-1))$.

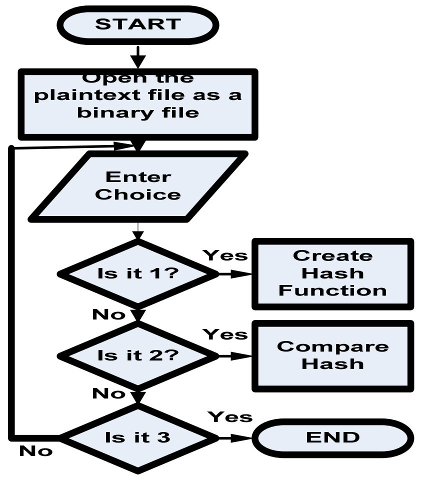
All these four steps of the key generation process can be performed easily. However, computing d from e and n appears very difficult. The Eavesdropper only knows n and so he can obtain p and q only by factoring$n$. This is believed to be a very hard and difficult task. Moreover, the only known way of breaking RSA by obtaining the decryption key d that, as we just argued, is difficult.

The designed RSA program implemented to obtain RSA key generation, encryption and decryption on demand. Fig.3 shows the RSA flowchart. It is divided into four parts. The first one generates two numbers and checks them whether they are prime or not. Then it executes arithmetic operations to find the public and private keys and save them together with the user ID to simple database file. The second part opens the request file as a binary file to achieve the encryption operation on it, using the equation needed for the public key (e). After that the program reads the file as four characters and so on until the file finished. The results are accumulated in the cipher file. The third part of RSA flowchart performs the decryption operation with the same equations. In this part the cipher file is either the IDEA session key or is Message Digest, which is encrypted by the private key. The results are accumulated in other file. The Fourth part manipulates the large lengths of the numbers that are used in this program. These lengths are variables; they could be up to 600 digits decimal number.

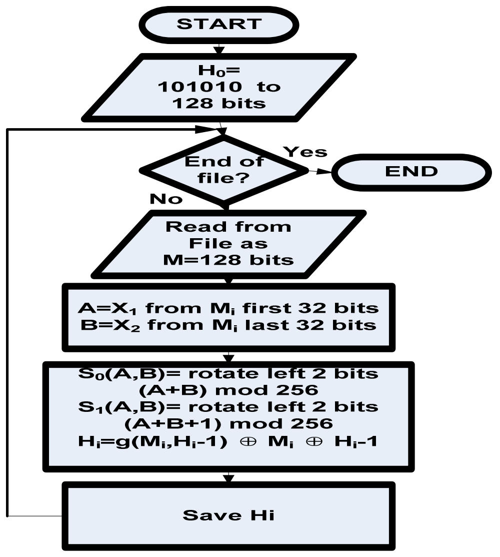
C. IDEA encryption operation

For IDEA encryption, the file is opened as a binary file and the program is designed to open all files such as (dat, txt,.avi, jpg,... etc.).

(a) Main options



(b) Create hash procedure


START
H_0= 101010 to 128 bits
End of file?
Yes
END
No
Read from File as M=128 bits
A=X_1 from M_i first 32 bits
B=X_2 from M_i last 32 bits
S_0(A,B)= rotate left 2 bits (A+B) mod 256
S_1(A,B)= rotate left 2 bits (A+B+1) mod 256
H_i=g(M_i,H_i-1) ⊕ M_i ⊕ H_i-1
Save Hi

(c) Compare two hash procedure

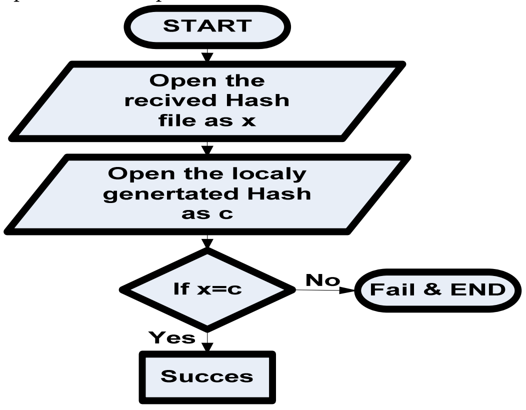

START
Open the recived Hash file as x
Open the localy genertated Hash as c
If x=c
No
Fail & END
Yes
Succes

Fig. 2 Hash Function procedure (a) main options (b) create hash (c) compare two hash

Then, the program generates a random numbers from the computer random number generator (RNG). After that, the program converts the decimal number of this random numbers to binary numbers. Then, the plaintext file is divided into blocks, each of them 64 bit and each block is divided into four sub-blocks X_1, X_2, X_3, X_4 .

The random number is also divided as $Z_1, Z_2, Z_3, \dots, Z_8$. Both X's and Z's are 16 bits. Now, the rounds are started to execute the arithmetic operations. Four sub-blocks become the input to the first round of the eight rounds of the algorithm. In each round the four sub-blocks are XORed, added and multiplied with one another and with six 16-bit sub-keys. Between rounds, the second and third sub-blocks are swapped.

Finally, the four sub-blocks are combined with four sub keys in an output transformation. In each round, the sequence of events is as follows:

(1) Multiply X1 by the first subkey.
(2) Add X2 to the second subkey.
(3) Add X3 to the third subkey.
(4) Multiply X4 by the fourth subkey.
(5) XOR the results of the steps (1) and (3).
(6) XOR the results of the steps (2) and (4).
(7)Multiply the results of step (5) with the fifth subkey Z_5.
(8) Add the results of steps (6) and (7).
(9)Multiply the results of step (8) with the sixth subkey Z_6.
(10) Add the results of steps (7) and (9).
(11) XOR the results of steps (1) and (9).
(12) XOR the results of steps (3) and (9).
(13) XOR the results of steps (2) and (10).
(14) XOR the results of steps (4) and (10).

The output of the round is the four sub-blocks that are the results of steps (11), (12), (13), and (14). Swap the two inner blocks (except for the last round) and that is the input to the next round. After the eighth round, there is a final output transformation:

(1) Multiply X1 and the first subkey.
(2) Add X2 and the second subkey
(3) Add X3 and the third subkey.
(4) Multiply X4 and the fourth subkey.

Finally, these four sub-blocks are attached to produce the ciphertexts. Creating the sub keys is also easy. The algorithm uses (52) of them (six for each of the eight rounds and four more for the output transformation). First, the 128-bit key Z is divided into eight 16-bit sub-keys. These are the first eight sub keys for the algorithm (the six for the first round, and the first two for the second round). Then, the key is rotated 25 bits to the left and again divided into eight sub keys. The first four are used in round 2; the last four are later used in round 3. The key is rotated another 25 bits to the left for the next eight sub keys and so on. The results are given from the variables $S_{15}, S_{16}, S_{17}, S_{18}$ after the eighth round has finished. These results are saved in a file. Then the rounds are repeated until the plaintext is finished.

The decryption part has the same steps and rounds, but the sub keys are reversed. The decryption sub keys are either additive or multiplicative inverses of the encryption sub keys. In this paper, for the purposes of IDEA, the all-zero sub-block (16 bits) is considered to represent $2^{16} = -1$ for multiplication modulo$2^{16} + 1$, thus the multiplicative inverse of 0 is 0. Calculating these take some doing, but can only have to do it once for each decryption key.

In the previous sections the operations are achieved without the knowledge of what was going on in the file. When the client selects the file that he wants to encrypt or decrypt by using the mouse pointer, he determines the destination to where he needs to send to, to server or other client. The implemented IDEA program is also configured to receive the key from the RNG when working in the encryption mode or from RSA output when working in the decryption mode, and the data from the ZIP depending on the operation mode. After the client selects the file, he can directly select the operation mode. These encryption operations are run sequentially in the sender side, as shown in the flowchart of Fig.4, and then select the recipient.

D. The sender Encryption Procedure

Fig.4 shows the overall secure files program flowchart at the sender side.

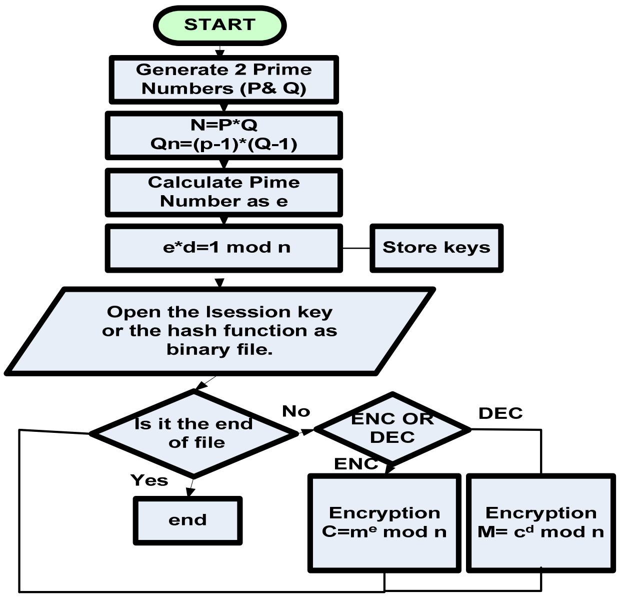


Fig. 3 RSA flow Chart

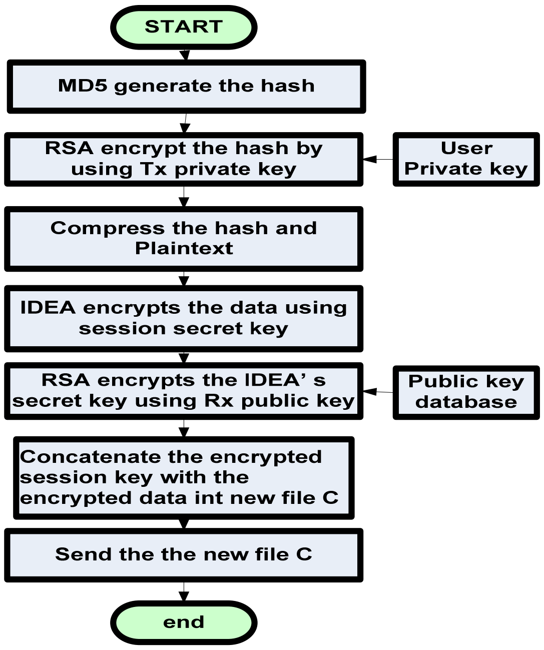

START
MD5 generate the hash
RSA encrypt the hash by using Tx private key
User Private key
Compress the hash and Plaintext
IDEA encrypts the data using session secret key
RSA encrypts the IDEA' s secret key using Rx public key
Public key database
Concatenate the encrypted session key with the encrypted data int new file C
Send the the new file C
end

Fig. 4 The flowchart of the overall procedure at the sender side

The flowchart of Fig. 4 explains the encryption procedures, which are explained in details in the previous sections. Therefore, the flowing will just describe the flowchart.

1. MD5 procedure generates the hash and then the digest is saved into file.

2. The RSA procedure generates the two keys (Public,

Secret) for the client and save them together with the user ID to the corresponding database file. Then, RSA encrypts the hash by using the client's private key.

3. The results of procedures (1, 2) and the plaintext file are merged together then, compressed using ZIP program.

4. The IDEA procedure generates 128-bit session key, the result of procedure (3) will be encrypted by the IDEA Procedure.

5. The session key is encrypted by RSA algorithm using the client public key.

6. Finally, the results in procedures (4, 5) above are merged and saved in temporary file in order to be sent.

The result of the program will be send as a temporary file (*.snd) to the server. The transmission process can be done by one of the flowing:

1. Simple way: cutting from the client's computer and then past it into special location in the file server. The encrypted file could be saved in the file server.

2. Copy file: copy the file to a share directory that can be available to the server or the destination client

3. Using the file transfer program, which use UDP or TCP protocols to send the file to server or other client through the network. However, many transfer techniques can be also used.

E. The receiver decryption procedure

The reviver decryption operation is shown in the flowchart of Fig. 5 and can be described as follows: -

1. After the file is received, the decryption program deals with the file as a sequence groups of 64-bit, these groups are spitted as follows:

• The first blocks are the encrypted data.

• The last two blocks are encrypted session key.

2. Decrypts the last two blocks using RSA program depending on recipient secret key to find the IDEA key.

3. The first blocks will be decrypted by IDEA program depending on the decrypted session key.

4. The results are accumulated as a temporary file and then it will be decompressed by ZIP program.

5. The resulted file is separated into two parts. The first one is the plaintext and the other is the message digest (hash).

6. MD5 compare the received hash with the locally generated one to check if any change is made to the plain file. This will obtain an integrity verification.

These points are done after the program checks the client's ID and his private key.

V. System analysis and Results

The proposed scheme of the secure file transfer system used in the design deal with the server as a normal client and as we see, the system is two way direction so, under this condition we can say the sender - whatever it can server or client - post his message to the recipient at any time. And the recipient always is waiting for the message to be accepted. However, the server can be used to store the public keys or as public key distributer. Therefore, the simple database table, which contains the user's public keys, can be saved in the server to be under access of all users. The system has been tested by sending several files using different scenarios, and a decent results is obtained. It is obvious that the security of this system greatly depend on the private key of the users. Therefore, this key needs to be securely stored by the user himself. In the implemented version, the RSA procedure automatically promotes the user to store his private key into a USB disk or secure server. In this paper, the length of the key is only 128 bits. However, the length of the keys and the extension of the files in the proposed system are all user-defined.

In this work, instead of using the plaintext data for authentication purposes, the authentication is obtained by hashing the real scanned signature file (bitmap file) of the user and encrypt and decrypt the hash using the private key of the sender D_B and the public key of the intended receipt E_s respectively. This feature is supported by the design system, because using the plaintext data for obtaining an authentication services may significantly open a vulnerability in the system that may alter the privacy service. That mean the intruder can use his knowledge of the public key and the reverse version of the hash function to restore the plain text. In this system, the hash function of the plain text is used only to obtain the integrity service and we can concatenate it with the plaintext information prior to be encrypted using IDEA.

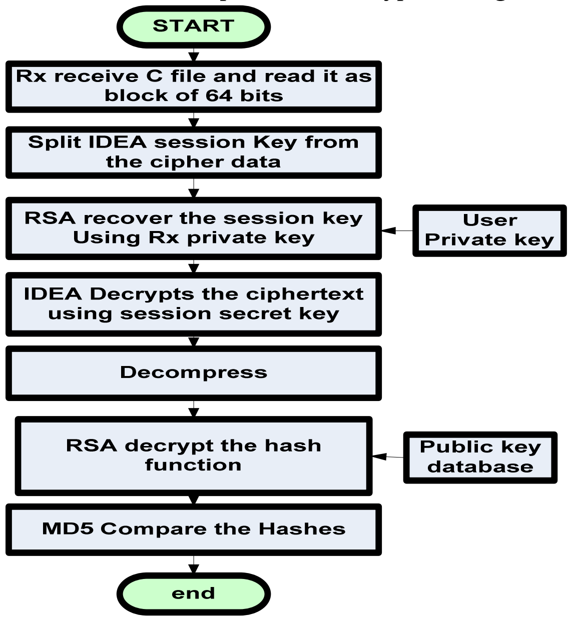

START
Rx receive C file and read it as block of 64 bits
Split IDEA session Key from the cipher data
RSA recover the session key Using Rx private key
User Private key
IDEA Decrypts the ciphertext using session secret key
Decompress
RSA decrypt the hash function
Public key database
MD5 Compare the Hashes
end

Fig. 5 The flowchart of the overall procedure at the receiver

On implementing of this system, it became necessary to have a controller to check the public keys and the private keys for each user together with hash file particularly in the case of a wide area network. The system creates a controller placed in the server to provide control on these keys. The proposed system is flexible and can be modified at any time if required. It includes algorithm and keys and can be utilized in Civilian establishments and Military Industry Companies. In this paper, we use the UDP protocol to send the file through the network also sending files can be done in LAN by cutting the file and then pasting it in a special location in the server as well as using real time communication system and social networks. In the internet, the encrypted file can be send with message as attachment in normal E-mail software. The users can gain many advantages such as, the protection of the sensitive information files. These files cannot be decrypted unless we have the private key of the intended user of these files and may be the program. In addition to protection against attackers, the program also protects the files from the viruses, because these files are not standard files (document, executable, etc.). Furthermore, for better utilization of the available bandwidth, the compressing of a file reduces its size and also increases its robustness against viruses and attacks.

Besides, RSA, which usually produces a bulk of data and has a very low speed, is only used in this system for the encrypting of the short key and hash function (only 128 bits). This is also very useful characteristic of the proposed system in the context of bandwidth utilization and speed. As the proposed system involves many types of keys in the transfer operation, it requires a public infrastructure to manage these keys. The proposed system simply solves this by creating a simple database file containing the public keys and this file can be published through the network. The database also contains the users IDs, their public keys and their additional public information. The secret key should be saved in a physically secured media by the user. One more things, the proposed system can be easily implement using a single chip to support other special two way onboard security systems and application.

VI. Conclusion

This paper proposed a secure file transfer system using the advantage of the modern cryptosystems. In one hand, the proposed system utilizes IDEA cryptography algorithm, which is more secure and have less implementation complexity, fast algorithm and very difficult to crack, for providing robust privacy services. In the other hands, as the key distribution is one of the common limitations of IDEA, the proposed system solves it by using one of the public key cryptography systems such as RSA. The proposed system can obtain higher authentication and integrity using the properties of the hash function and RSA. The proposed system can be updated by using other symmetric and asymmetric cryptosystems instead of IDEA and RSA.

ACKNOWLEDGMENT

This work is supported by National Natural Science Foundation of China (No. 610011024) and the National Key Technology R&D Program under the grant 2012BAH15B00.

References

[1] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, 1990, pp. 296-304.

[2] C. Kaufman, R. Perlman, and M. Speciner, *Network security: private communication in a public world*: Prentice Hall Press, 2002.

[3] A. Abdelgader and L. Wu, "A Secret Key Extraction Technique Applied in Vehicular Networks," in *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*, 2014, pp. 1396-1403.

[4] S. N. Kumar, "Review on Network Security and Cryptography," *Science and Education*, vol. 3, pp. 1-11, 2015.

[5] J. Katz and Y. Lindell, *Introduction to modern cryptography*: CRC Press, 2014.

[6] D. Bailey, M. Campagna, R. Dugal, and D. McGrew, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS," 2014.

[7] T. Herath, R. Chen, J. Wang, K. Banjara, J. Wilbur, and H. R. Rao, "Security services as coping mechanisms: an investigation into user intention to adopt an email authentication service," *Information Systems Journal*, vol. 24, pp. 61-84, 2014.

[8] K. C. Silvester, "Using a Communication Protocol to Provide Security Services," ed: Google Patents, 2014.

[9] R. Amin and G. Biswas, "Anonymity preserving secure hash function based authentication scheme for consumer USB mass storage device," in *Computer, Communication, Control and Information Technology (C3IT), 2015 Third International Conference on*, 2015, pp. 1-6.

[10] S. Chandra, S. Paira, S. S. Alam, and G. Sanyal, "A comparative survey of Symmetric and Asymmetric Key Cryptography," in *Electronics, Communication and Computational Engineering (ICECCE), 2014 International Conference on*, 2014, pp. 83-93.

[11] A. Gupta, R. Mittal, and G. RKGITW, "Symmetric and Asymmetric Cryptosystem," 2014.

[12] N. Nassar, R. Newhook, and G. Miller, "Enhanced mobile security using SIM encryption," in *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, 2014, pp. 189-196.

[13] H. B. Pethe and S. Pande, "A Survey on Different Secret Key Cryptographic Algorithms," *IBMRD's Journal of Management & Research*, vol. 3, pp. 142-150, 2014.

[14] S. Kumari and J. Chawla, "Comparative Analysis on Different Parameters of Encryption Algorithms for Information Security," 2015.

[15] S. G. Krantz and H. R. Parks, "RSA Encryption," in *A Mathematical Odyssey*, ed: Springer, 2014, pp. 197-215.

[16] S. Challenge, J. Zhang, D. Zhang, and A. Drew, "Number Theory Applied to RSA Encryption," 2015.