ORIENTAL JOURNAL OF
COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
March 2017,
Vol. 10, No. (1):
Pgs. 24-32

Access Control and Encryption of Sensitive Data Using i-Se4GE Algorithm

DR. MAMTA PADOLE and PRATIK KANANI*

Department of Computer Science & Engineering,
Maharaja Sayajirao University of Baroda, Baroda, India.
Department of Information Technology, University of Mumbai,Mumbai, India.
*Corresponding author E-mail: Pratikkanani123@gmail.com

http://dx.doi.org/10.13005/ojcst/10.01.04

(Received: January 19; Accepted: February 08, 2017)

ABSTRACT

Encryption, itself doesn't prevent interception, but denies the message content to the interceptor. For technical reasons, an encryption scheme usually uses a pseudo-random encryption key generated by an algorithm. It is impossible to decrypt the message without possessing the key, but, for a well-designed encryption scheme, large computational resources and skills are required to decrypt it. An authorized recipient can easily decrypt the message with the key provided by the originator to recipients, but not to unauthorized interceptors.

Data hiding is the skill of hiding messages in such a ways the only the sender and the receiver of the message knows that the message has been hidden. In the context of secured information transmission and reception, efficient techniques for data encryption and decryption are very much essential. In this paper, a message transfer application is developed which mainly focusses on secret message transfer between two parties. Used scheme not only focusses on text but also the encryption of images and videos. A unique algorithm i-Se4GE is used for the transfer of keys required for the encryption and decryption of messages. This algorithm makes use of a set of random numbers, timestamp of the two users, a set of public keys and dynamic keys. The algorithm uses a two-step authentication request-reply process which provides it a double layer security.

Keywords: i-Se4GE, Cryptocat, TestSecure, MME, HS, Attacks.

INTRODUCTION

Transferring of Text/Audio/Video through an application by using cryptography is the software aimed at providing a high level of security to the sensitive data for organizations, since they face issue in using the existing system in which they cannot send all three media files. The aim is all about security and making the sensitive data secured. There is an additional feature in the project of self-liquidation according to which the data gets deleted after a fixed amount of time.

The user has an option to send audio, video or text. On successful decryption using the primary key doesn't guarantee access to the data, it depends on the privileges of the receiver.

The goal of our project is to make data unreadable by a third party. The goal is to hide the data from a third party with the use access control policies and encryption algorithms allowing them to make the sending of data secure. Together, these two are intended to fight unauthorized access to the data. Also if the encryption algorithms fails to be executed in the application then the message is not sent to the server and an error is provided to retry the process[1,2].

The message application developed can be used for military purposes as the messages will be encrypted which is needed for every organization to communicate. We have added the access control policies which help data delivered only to the desired user. Other than the military, it can be used by any external parties to send confidential messages. It can also be used to share images and videos that are meant to be sensitive.

Background And Related Work

Existing systems are identified and studied to develop an application which will overcome the disadvantages .

Cryptocat[3,4]

One of the open source mobile and web application tool which allows secure and encrypted chatting is Cryptocat. Cryptocat uses end-to-end encryption and encrypts chats on the client side, only server trust with data that is already encrypted. Cryptocat is existing as an app for Mac OS X or as a browser extension for Google Chrome, Mozilla Firefox, Apple Safari, Opera and as a mobile app for iPhone. Cryptocat's defined goal is to make encrypted communications more accessible to average users. The chat software aims to strike a balance between security and usability recommending more privacy than services such as Google Talk or Internet Relay Chat, while maintaining a higher level of accessibility than Pidgin. Cryptocat is developed by the Cryptocat team and is released under the GPLv3 license. Cryptocat for the iPhone works pretty much the same as the desktop counterparts. The process to use Cryptocat is pretty simple: create a private chartroom with a unique name share that name with your friend over another form of communication like text message, then they'll create a one-time use username and enter the chat room. From there, you can chat away about whatever you want, and it won't save any of it after you egress the app. But, can only encrypt messages between cryptocat users and app does not mask IP address.

TextSecure [5,6]

TextSecure is an cutting-edge end-to-end encryption protocol as well as a free and open-source encrypted instant messaging application for Android which uses that protocol. TextSecure enables the secure transmission of instant messages, group messages, attachments and media messages to other Text Secure users. Users can independently verify the identity of their correspondents by relating key fingerprints out-of-band or by scanning QR codes in person. The Android application can function as a drop-in replacement for Android's native messaging application as it can also fall back to sending unencrypted SMS and MMS messages. The local message database can be encrypted with a passphrase. TextSecure messages are compatible with Signal messages on iOS. TextSecure and Signal are developed by Open Whisper Systems and are published under the GPLv3 license. TextSecure allows users to send encrypted text messages, audio messages, photos, videos, contact information, and a wide selection of emotions over a data connection (e.g. Wi-Fi, 3G or 4G) to other TextSecure users with smartphones running Android and to Signal users on iOS. TextSecure also allows users to exchange unencrypted SMS and MMS messages with people who do not have TextSecure or Signal. Messages sent with TextSecure to other TextSecure users and to Signal users are automatically end-to-end encrypted, which means that they can only be read by the intended recipients. The keys that are used to encrypt the user's messages are stored on the device alone, and they are secured by an additional layer of encryption if the user has a passphrase enabled. In the user interface, encrypted messages are denoted by a lock icon. Sometimes it has trouble

sending images/videos/audio clips and if a user's password is cached in the phone, new texts will appear in plain text in the notification center.

Pitfalls In Existing Systems[3,5,7-9]

1) Sharing the key: The biggest problem with symmetric key encryption is that you need to have a way to get the key to the party with whom you are sharing data. Encryption keys aren't simple strings of text like passwords. They are essentially blocks of garbage. As such, you'll need to have a safe way to get the key to the other party. Of course, if you have a safe way to share the key, you probably don't need to be using encryption in the first place.

2) More Damage is compromised: When someone gets their hands on a symmetric key, they can decrypt everything encrypted with that key. When you are using symmetric encryption for two-way communications, this means that both sides of the conversation get compromised. With asymmetrical public-key encryption, someone that gets your private key can decrypt messages sent to you, but can't decrypt what you send to the other party, since that is encrypted with a different key pair.

3) Digital Signatures: One major disadvantage of the symmetric encryption is that it cannot provide digital signatures that cannot be repudiated.

4) Speed: A disadvantage of using public-key cryptography for encryption is speed. There are accepted secret-key encryption methods which are significantly faster than any current available public-key encryption methods.

i-Se4ge Algorithm[13]

The aim is to create an application that encrypts all the media files before being sent to the receiver. Since, there has been a great increase in the amount of data available on the internet due to which it is important to encrypt and send the data which is critical to an organization or a company.

With the Increase in the computing power it also increases the security risk. With more speed and powerful processors, the security cracking mechanism also becomes powerful. Therefore to overcome such limitations there is a strong need of multiple and long bit key lengths. The Attacker keeps an eye on the data flowing through the network, captures the data and tries to crack it by different means of tools and techniques. For example DoS, replay attack, forgery attack, fabrication, interception, interruption, modification and man-in-the-middle attack. The wide range and availability of wireless network makes it more vulnerable than the wired network. To solve this problem, the improved Security system with RSA and Diffie-Hellman algorithms for 4G Environments (i-Se4GE) was proposed for 4G LTE networks. This algorithm makes use of a set of random numbers, timestamp of the two users, a set of public keys and dynamic keys. The algorithm uses a two-step authentication request-reply process which provides it a double layer security. Related terms are as follows which helps to understand the i-Se4GE algorithm.

The traditional network core is replaced by Evolved Packet Core in LTE-A (4G) networks. The core elements of LTE-A networks are as follows[10,11,12]

1. **Evolved Packet core** *:* The EPC is fully IP based high speed packet switched backbone network.
2. **User Equipment** *:* UE provides User Interface to end users and contain Universal Subscriber Identity Module (USIM). USIM is used to identify and authenticate the user.
3. **E-UTRAN Node Bs** *:* eNB is a radio station, distributed throughout the networks to increase the coverage area. Radio Network Controller (RNC) is implemented inside eNB responsible for header compression, ciphering and reliable packets delivery with radio resource management.
4. **Mobility Management Entity** *:* MME manages users mobility and performs authentication and authorization. It stores UE status with network architecture specific signaling.
5. **Home Subscriber Server** *:* HSS is the central database server maintained in user operators premises. It contains list of services authorized for users. Also it has DCC, that contains a set of secret keys for every user.

Most of the functions and terms used in i-Se4GE are same as Se4GE, but based on the

need of new system some modifications have been made. The Random Number Generator (RNG) is used in proposed system architecture as they possess characteristics such as unpredictability, randomness, uniformity, independence, stability, initial sensitivity value and long repeating periods.

Secured DCC and System Parameters

Based on users IMSI, MME retrieves DCC from HSS and checks whether the message is sent by legal user or not. In proposed scheme DCC contains IMSI, e_i, d_i, N_i and K_i. But all these keys are encrypted by G_k, which is possessed by MME only. So if a hacker obtains users DCC by any mean, he will not able to get user set of keys.

Encryption: Cipher = Text $\oplus$ G_K
Decryption: Text = Cipher $\oplus$ G_K

These set of keys are generated by AAA server and stored on both sides when user registers itself for the first time.

i-Se4GE parameters are as follows:
1) p and q : strong prime numbers present on UE and MME side to generate keys of desirable lengths, where q is next probable prime of p.
2) IMSI : International Mobile Subscriber Identity, is a combination of Mobile Country Code (MCC), Mobile Network Code (MNC), and Mobile Subscriber Identification Number (MSIN).
3) K_i ,e_i , d_i and N_i : the encrypted keys present in DCC where (e_i , d_i and N_i) is RSA triplet and K_i is an user authentication key.
4) T_{UE} , T_{MME} and ΔT : timestamp of UE, MME and their difference respectively.
5) $AK_1 \sim AK_5$: $AK_{1\sim3}$ and $AK_{4\sim5}$ are random numbers of desired length used as authentication keys by MME and UE respectively.
6) $MR_1 \sim MR_3$: are random numbers of desired length generated each time, whenever user tries to communicate with MME. So that every time user has new set of dynamic keys. This is to prevent attacks based on previous session cracked keys if any. They are used as user equipment private keys.
7) $BR_1 \sim BR_2$: MME Private keys as UE has.
8) P_{MRj} : $P_{MRj} = q^{MRj} \bmod p$, $1 \le j \le 3$: user equipment public keys.
9) P_{BRj} : $P_{BRj} = q^{BRj} \bmod p$, $1 \le j \le 2$: MME public keys.
10) $CSK_j = (P_{MRj})^{BRj} \bmod p$, $1 \le j \le 2$: common secret keys generated by MME.
11) $DK_{1\sim12}$, $DA_{1\sim4}$, $DB_{1\sim4}$: Dynamic Keys which are randomly generated on both sides to support protection key chain principle.
12) $TEK_{0\sim47}$ and $NTEK_{0\sim47}$: are the set of encryption keys used to encrypt and decrypt the data messages.

The i-Se4GE Functions

The Logical functions used in Algorithms are as follows.
1) $U +_2 V$: where $+_2$ indicates Logical OR operation which ignores most significant carry.
2) $X \oplus Y$: where $\oplus$ represents Logical Ex-OR function.
3) Dafun(a,b,c) = $(a \oplus b) + c$: is data authencation function.
4) IDafun(a,b,c) : inverse of Dafun() is defined as

$$a = IDafun(a,b,c) = \begin{cases} (x - c) \oplus b, & \text{if } x \text{e" } c \\ (x + c + 1) \oplus b, & \text{if } x < c \end{cases}$$

Where x= Dafun(a,b,c). The IDafun() enables retrieval of *a* if and only if the user has b and c. This way it makes an intelligent protection key chain and guarantees that user will get *a* only if he has b and c.
5) $RSA\_En(m_i) = (m_i)^{ei} \bmod N_i$: encrypts plain text message expressed as BigInteger.
6) $RSA\_De(c_i) = (c_i)^{di} \bmod N_i$: decrypts cipher text to plain text expresses as BigInteger.
7) HMAC(k) : Hash based Message Authentication Code. Ensures the integrity of sending and receiving message.

The set of Operational Codes are used to get the status of the system. With the help of the given Op_codes the system understands the message format and checks for it, based on the received Op_codes the system performs the operations given in algorithms. For all messages system uses a particular Op_codes and if they are not matched the system discards the process.

The list of Op_codes followed in i-Se4GE is as follows.

Table 1: Defined Op_codes

Op_ code	Process	Explanation
1	First Authentication Request	By UE to MME
2	First Authentication Reply	By MME to UE
3	Second Authentication Request	By UE to MME
4	Second Authentication Reply	By MME to UE
5	Data Transmission	From UE to MME
6	Data Transmission	From MME to UE
0, 7-15	Reserved	For future use

Fig. 1. represents the overall key management process of the proposed system.

i-Se4GE Algorithm

Step 1: With the help of eNB, UE connects to the MME. Sets Op_code as 1 and operation as first authentication request. The format of the message is as follows.

$$OP_{code} \mid T_{UE} \mid *IMSI \mid RSA\_En(P_{MR1}) \mid RSA\_En(P_{MR2}) \mid HMAC((IMSI +_2 P_{MR1}) \oplus P_{MR2})$$

In this message UE sends, its own timestamp to MME. Different users connects to MME by different network ids. MME knows which network carries what data, so * indicates IMSI encryption by respective network ids to make it more secure. Such schemes can be applied to eNBs.

Step 2: When MME receives Message 1, based on the Op_code it gets the system state, finds T_{MME} and then it calculates ΔT by $T_{MME} - T_{UE}$. If ΔT is more than expected, then MME discards the communication process, else it finds IMSI and requests HSS to provide users DCC. After decrypting DCC keys, MME finds P_{MR1} and P_{MR2}.

Now it checks the message for its integrity by comparing $HMAC((IMSI +_2 P_{MR1}) \oplus P_{MR2})_r$ and $HMAC((IMSI +_2 P_{MR1}) \oplus P_{MR2})_c$? Where $_r$ and $_c$ represents received and calculated HMAC respectively. If $HMAC_r = HMAC_c$ then it generates $AK_{1\sim3}$ and sends authentication reply to UE else discard the process.

Step 3: MME sets Op_code value as 2 and computes $CSK_{1\sim2}$.

The format of first authentication reply is as follows.

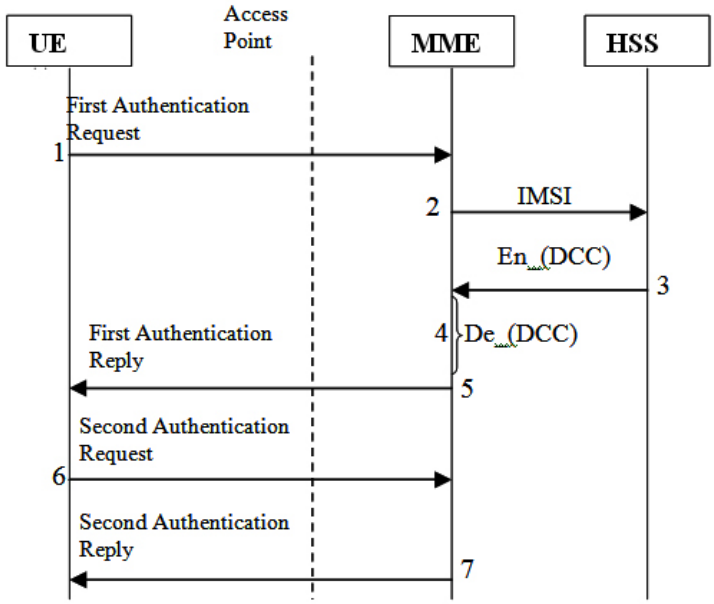

Fig. 1. Sequence diagram for i-Se4GE algorithm

$Op\_code \mid Dafun(P_{BR1}, P_{MR1}, IMSI) \mid Dafun(CSK_1, P_{BR1}, P_{MR1}) \mid Dafun(CSK_2, CSK_1, P_{BR1}) \mid Dafun(AK_1, CSK_1, CSK_2) \mid Dafun(AK_2, AK_1, CSK_2) \mid Dafun(AK_3, AK_2, AK_1) \mid HMAC((P_{BR1} \oplus CSK_2) +_2 AK_3)$

Here MME uses a Protection key chain mechanism. Here UE has P_{MR1} and IMSI value , so only legitimate UE can retrieve the P_{BR1} value by using IDafun(). In next Dafun(), MME sends CSK_1 which can be retrieved by UE if and only if it has found correct IDafun() for first Dafun(). Similarly next Dafun() depends on previous IDafun().

MME generates 12 dynamic keys as $DK_{1\sim12}$ by using the given formula.

$DK_{(j-1)*6+(K-1)*3+1} = [P_{BR1} +_2 (P_{MRj} \oplus CSK_k)] \oplus AK_l$, $1 \le (j,k) \le 2$ and $1 \le 1 \le 3$.

Step 4: After receiving the message UE checks for the Op_code and it understands that the message is first authentication reply. By using P_{MR1} and IMSI it finds P_{BR1}. Then by using P_{BR1} it does IDafun(CSK_1, P_{BR1},P_{MR1}) and finds CSK_1. Likewise UE finds CSK_2, AK_1, AK_2, AK_3 and checks for $HMAC((P_{BR1} \oplus CSK_2) +_2 AK_3)_r$ and $HMAC((P_{BR1} \oplus CSK_2) +_2 AK_3)_c$. If they are not equal then discard the process else continue. Now based on the received key values UE also generates $DK_{1\sim12}$.

Step 5: UE sets Op_code as 3 and prepares for second authentication request. The message format is

$Op\_code \mid RSA\_En(P_{MR3}) \mid Dafun(AK_4, DK_5, DK_7) \mid Dafun(AK_5, P_{MR3}, AK_4) \mid HMAC((AK_5 \bullet" P_{MR3}) + AK_4)$

Step 6: After receiving the message MME checks for the respective Op_code. If it is desirable then it continues else discards the message. MME decrypts $RSA\_En(P_{MR3})$ function to obtain P_{MR3} and it finds IDafun() for first and second Dafun() to get AK_4 and AK_5.

After finding required keys, it checks for authenticity of $HMAC_r$ and $HMAC_c$. If the authentication is true, it continues else discards the message and waits for the arrival of true message.

MME generates $DA_{1\sim4}$, $DB_{1\sim4}$, $TEK_{0\sim47}$, *and* $NTEK_{0\sim47}$, by using the formula given,

$DA_i = (P_{MR3} +_2 AK_i) \oplus (P_{BR2} +_2 AK_j)$, $1 \le i \le 4$, $j=i+1$.
$DB_i = (P_{BR2} \oplus AK_j) +_2 (P_{MR3} \oplus AK_i)$, $2 \le j \le 5$, $i=j-1$.
$TEK_{(i-1)*4+(j-1)} = DK_i \oplus DA_j$, $1 \le i \le 12$, $1 \le j \le 4$.
$NTEK_{(i-1)*4+(j-1)} = DK_i +_2 DB_j$, $1 \le i \le 12$, $1 \le j \le 4$.

Step 7: MME sends second authentication reply to UE as

$$Op\_code \mid Dafun(P_{BR2}, AK_4, AK_5) \mid HMAC(P_{BR2} +_2 AK_4)$$

In this step MME gives its second public key P_{BR2} to UE, so that it can calculate the set of keys calculated by MME in step 6.

Step 8: From the Op_code UE knows that it is second authentication reply and UE obtains P_{BR2} from Dafun(P_{BR2}, AK_4,AK_5) by invoking IDafun(). Then UE checks for the received and calculated HMAC function. After authentication UE generates $DA_{1\sim4}$, $DB_{1\sim4}$, $TEK_{0\sim47}$, $NTEK_{0\sim47}$. After the step 8 is over both UE and MME has the set of required keys needed for encryption and decryption of the messages.

Step 9: UE sends first set of cipher texts and its corresponding Op_code to MME, where Plaintext = P_0 , P_1 , P_2 , P_3 , ... , P_{n-1} and corresponding Ciphertext = C_0 , C_1 , C_2 , C_3 , ... , C_{n-1}.

$C_i = (P_i \oplus TEK_x) +_2 NTEK_y$, $0 \le i \le n-1$, $x=i+1$, $y=i+2$.

If x and y goes beyond 47 then use mod 47 operator to bring them in range. The message format for step 9 is as follows.

$$Op\_code \mid Ciphertexts$$

Here each cipher text is generated by the set different TEK_s and $NTEK_s$ from the list. So it becomes very much difficult to find such combination of keys. Even to decrypt these messages, having the keys are not so enough, but the order of the keys are also important.

Step 10 : Based on the received Op_code, MME sets its own Op_code and decrypts the message to obtain plaintexts as follows.

$$P_i = \begin{cases} (C_i - NTEK_y) \oplus TEK_x, & \text{if } C_i \geq NTEK_y \\ (C_i + NTEK_y + 1) \oplus TEK_x, & \text{if } C_i < NTEK_y \end{cases}$$

Where $0 \leq i \leq n-1$, $x=i+1$, $y=i+2$. If x and y goes beyond 47 then use mod 47 operator to bring them in desired key range.

Security Analysis

Security analysis of i-Se4GE is done to check its vulnerability against major attacks[4,8].

1) **Replay attack** : The system uses T_{UE} and T_{MME} which are timestamps of UE and MME respectively. ΔT is measured, based on the average network delay between UE and MME. So, if attacker captures the message, modifies it and sends it back to MME than ΔT exceeds the expected value and the key management process is discarded. Hence system provides security against replay attack.
2) **Forgery attack** : System uses DCC mechanism without which the keys cannot be decrypted further and DCC is also encrypted. This makes it unusable for attacker even if he owns it by some mean. Also on both sides, Dafun()s are used which provides a Protection Key chain between trusted parties. Hence the system is safe against forgery attack.
3) **DoS** : Proposed system is protected against DoS attacks, since without the set of proper Op_codes and message formats. MME will not entertain any messages, hence forgeable messages will be discarded.
4) **Evesdropping attack** : Even if hacker cracks one or more keys and acquires DCC, still the attack is not possible because the system is using multiple keys which makes it impossible to crack all keys with their particular order and combinations.
5) **Statistical attack** : The system uses multiple and long bit random keys generated by RNG which provides security against statistical attack.
6) **Birthday attack** : The HMAC function used are finding the hash based on the $+_2$, $\oplus$, and + of the keys. So even if attakcer cracks it, he will get the result as some value of key operations but not the actual keys.
7) **Brute force attack** : For few combinations the attacker may get desirable permutation and combinations, but the system uses so many keys and their order is also important. This makes brute force attack impossible.
8) **Time interval attacks** : Every time when UE prompts to MME the system generates new set of private and public keys. So even if attacker captures all messages, cracks it and after some time period tries to get into the session he will not be able to do so.

Implementation of Text Securely

This project not only focuses on encrypting text but also on the encryption of images and videos. Our application can encrypt all media files. We are using "**Blowfish**" algorithm for the encryption and decryption of media files[14].

Working Flow

1. Start
2. Browse and Select the File to be encrypted and sent.
3. Click on the Send Button.
4. Key-Generation Algorithm starts.(For Key-Generation Algorithm refer to)
5. Key generated and verified with the receiver
6. Authentication Completed.
7. Encrypted data is sent to the receiver.
8. On clicking on decrypt button the data is decrypted.
9. Complete

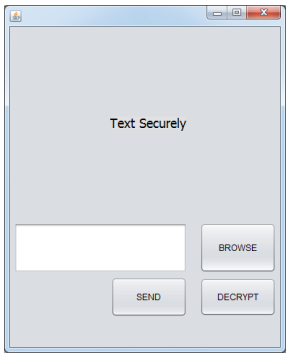

Fig. 2: Media to be encrypted

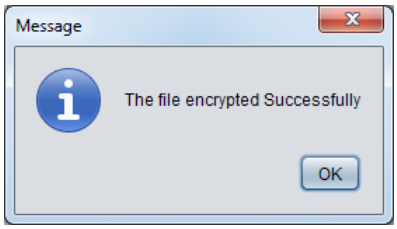

Fig. 3. Media Encrypted Successfully

User Interfaces

1) Encryption process
2) Decryption Process

CONCLUSION

Text Securely is an important encryption tool. Attacks such as man-in-the-middle and Denial-of-Service can be reduced considerably by using our key generation algorithm. This application ensures that the data is encrypted and prevents unauthorized users from gaining access to the data since the keys are changed for every new communication link that is established. Thus, a Web based chat application that can be useful for organization to send sensitive data over a secure network is implemented which encrypts and decrypts any files irrespective to its extensions.

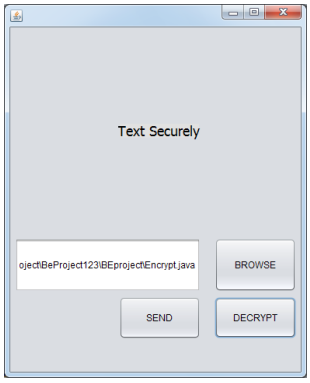

Fig. 4: Media to be dencrypted

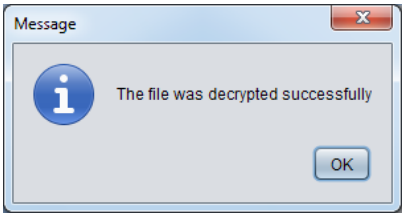

Fig. 5: Media dencrypted successfully

REFERENCES

1. S. Sharma, J. Bisht, "Performance Analysis of Data Encryption Algorithms," *IJSROSET*, **3**(1), 1-5, 2015.
2. Jacobs School of Engineering. (2014,January,10). Computer Science and Engineering. [Online]. Available :https://cseweb.ucsd.edu/~mihir/cse207/w-asym.pdf
3. Cryptocat. (2016,Feb,16). Cryptocat. [Online]. Available: https://crypto.cat/
4. True private Messaging . (2016, March, 20). [Online], Available : http://www.whoishostingthis.com/blog/2015/04/29/im-encryption/
5. Security-in-a-Box . (2016, March, 25). [Online], Available : https://securityinabox.org/en/guide/textsecure/android
6. How Secure is Text Secure? . (2016, March, 20). [Online], Available : https://eprint.iacr.org/2014/904.pdf
7. Naked Security . (2016, March, 27). [Online], Available : https://nakedsecurity.sophos.com/2013/07/06/cryptocat-encrypted-group-chats-may-have-been-crackable-for-7-months/
8. Health BI . (2016, March, 28). [Online], Available : http://www.healthbi.com/secure-messaging-careful-pitfalls/
9. Schneier on Security (2016, March, 29). [Online], Available : https://www.schneier.com/essays/archives/1998/01/security_pitfalls_in.html
10. J. Cao and M. Ma, "A Survey on Security Aspects for LTE and LTE-A Networks," IEEE, *Communication Surveys and Tutorials*, **16**(1), 283-302, 1955.
11. R. Nossenson, "Long-Term Evolution Network Architecture," IEEE, International

conference on Microwave, Communications, Antennas and Electronics Systems, 2009, pp.1-4.

12. Y. Huang, F. Leu and J. Liu, "A Secure Wireless Communication System Integrating PRNG and Diffie-Hellman PKDS by Using a Data Connection Core," IEEE International conference on Broadband Wireless Computing, Communication and Applications, 2013, 360-365.
13. P.Kanani, V.Kaul, K.Shah," Hybrid PKDS in 4G" ,ICSPCT 2014, pp- 323-328.
14. Java Cryptography Architecture standard algorithm name documentation. (2016,May,2015). [Online]. Available: http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html#CipherM.Young, *The Technical Writer's Handbook*, Mill Valley, CA: University Science, 1989.