Implementation of EAX Mode of Operation within a Real-Time Android Chatting Application

Levent Ertaul, Nikhitha Vadla Konda, Dharani G Ramasamy
CSU East Bay, Hayward, CA, USA
levent.ertaul@csueastbay.edu, nvadlakonda2@horizon.csueastbay.edu,
dgobichettypalayamramasamy@horizon.csueastbay.edu

***Abstract*— SMS has been a widely used data application till today, availing two-end users to share any information traditionally. Android chatting application helps the users to send and receive textual messages to and from other android mobile devices. Secrecy of messages was achieved through encryption but authentication was still pertain and the information shared was prone to several attacks either be eavesdropped or say identity theft. In order to facilitate a secure environment, this paper proposes one of the block-cipher modes of operation, EAX. The primary objective of this application mainly accounts on the implementation of EAX mode of operation within a real-time chatting application to ensure authenticated encryption. The JAVA programming language is used to implement the application with PHP language at the server-side connected to the SQLYog, a GUI tool for MySQL database that runs on the tomcat server. Finally, the performance analysis is calculated to demonstrate the efficiency and suggestions are made to improve the message security in the android application**.

I. INTRODUCTION

Nowadays, there are many numbers of android mobile applications that allow users to send and receive messages and share confidential information with others. There are many possibilities for it to be vulnerable when sharing the confidential data via text messages [1]. Encryption, decryption and authentication are the key technologies which help to provide security for the information. At the same time android mobile message service providers need more attention to avoid data theft and other illegal activities in the application while sharing confidential data. EAX provides integrity, confidentiality of the messages being shared with authenticity. By availing the EAX technology in the messaging service helps to avoid these vulnerabilities. Because breaking EAX security is one of the hardest process and anonymous users or hackers need more time and many number of password combination to break EAX security.

Authentication Encryption (AE) schemes are based on the symmetric key mechanisms and message M is converted into cipher-text C, this cipher-text will help to protect both privacy and authenticity of message [2]. Naïve combinations are implemented using the basic AE scheme for existing message authentication code (MAC) encryption. EAX mode is an algorithm and it is used as cryptographic block cipher to pledge an information service such as authenticity in the system. Commonly it is designed to use for both authentication and privacy for the messages with two pass scheme for securing data [3]. One pass is used for achieving privacy and one for authenticity for each block in the message sending and receiving. At present, EAX is known as popular encryption and authentication mode of algorithm for terminal equipment. CTR (Counter Mode) and OMAC (One-Key CBC MAC) mechanisms have been used in this EAX encryption to enrich data security while transferring data through internet [2]. The proposed android secret application is implemented with EAX technique to provide enriched security for the user's messages.

Anyhow, EAX encryption and authentication mode includes CTR (Counter Encryption mode) encryption mode and OMAC (One-Key CBC MAC) authentication mode to create high security for the message. [4]. Comparing the CCM mode speed to encrypt and decrypt data, EAX is quicker than CCM. Algorithm complexity of the EAX is lower than other encryption algorithms such as CWC and GCM and the software and hardware resources that need to run this algorithm is low, so encryption and authentication of EAX mode is very quick comparing with other algorithms [5].

OMAC authentication provides some adaptation policy to CBC mode. CBC provides a feedback loop which enables high security for the encryption algorithm and it is iterative and cannot be process parallel data. So it cannot be maximize the processing speed [6].This research work is mainly concentrated on creating an android application which provides functionality to the user to send and receive message to other users. This communication process is encrypted using EAX mode to provide high security for the message. The detailed information about the proposed android application will be discussed in the below section [7].

Section II offers, brief description of EAX, CTR and OMAC algorithms and accounts in-depth working of encryption and decryption process in these algorithms. Section III provides implementation process of the android chatting application and also it explains architecture of the android application and database that is used to store application data. This section covers almost seventy percent of the android chat application functionalities and other processes. Section IV describes about the android application performance analysis and sample java implementation code of the application. Finally, section V concludes about the use of EAX algorithm the proposed android chat application.

II. EAX MODE OF OPERATION

For the cryptographic block cipher, EAX mode is a mode of operation and it is an Authenticated Encryption with Associated Data (AEAD) algorithm. It is designed to provide privacy and authentication for the message over the network by achieving authenticity and achieving privacy [16]. EAX is a mainly a nonce based AEAD scheme which use block cipher. EAX is a two pass AEAD scheme, by using no known intellectual property it separates authenticity pass and privacy pass [17]. EAX provides its functionality in more flexible and supports the arbitrary length message. Complicated length annotations are avoided by the EAX [18].

EAX always use only the forward direction of bock cipher for both encryption and decryption and so there is no need of separate decryption functionality of block cipher to get implemented. EAX is more secured by using of its block cipher by using with pseudo random number. Main securities in the EAX are authenticity of cipher text and its distinctive random bits[19].In our android chatting application, EAX is mainly used to enforce a secure environment between the two-end users in sharing the data over the network. EAX is having the efficient security features for both authentication and encryption and so this is included in our application [20].

1) *EAX Algorithms*

EAX algorithms fix a block cipher with tag length and key, these are the two main parameters which are fixed in beginning of the EAX mode of operation. There parameter are agreed between sender and receiver in an authenticated manner. EAX provides nonce based AEAD scheme, where the nonce use these two parameters key and tag length, encryption is carried on by using the nonce and encryption algorithm use the signature are Nonce× Key× Header× Plaintext -> Cipher text and decryption algorithm has signature as Nonce× Key× Header× Cipher text -> Plaintext. Nonce, key, header are sequence of binary numbers or random numbers [21].

EAX algorithm does not consist of any encoding of multiple strings and it converts the multiple strings into single one. Pseudo Random Numbers are used for generating nonce, key and header. These multiple strings are converted into single one in simple, compelling, on-line and efficient way. EAX avoid encodings to increase the standard of the encryption and all the parameters use Pseudo random number to increase the complexity of the algorithm by increasing its security features [22].

2) *EAX Encryption*

Encryption procedure in the android chat application will process as given in the Figure 1. The step by step process of Encryption and authentication is as follow,

Step 1: N $\leftarrow$ OMAC^0_K (N), Initial value is Zero, with the OMAC mode nonce is authenticated, N is the result of authentication and this is used as CTR initial vector.

Step 2: H $\leftarrow$ OMAC^1_K (H), Initial value is one, with OMAC mode the header message is authenticated and the H is the result of this process

Step 3: C $\leftarrow$ CTR^N_K (M), Initial value is N, with CTR mode message is authenticated and the C is the cryptography after the encryption.

Step 4: e $\leftarrow$ OMAC^2_K (C), initial value is two, with OMAC mode the cryptography C is authenticated and the e is the result of this process.

Step 5: Tag$\leftarrow$ N (XOR) e (XOR) H, Integrated authentication sign tag is created in this process.

Step 6: T $\leftarrow$ Tag |first r bits|, final authentication sign T is produced in this process.

Step 7: return CT $\leftarrow$ C || T, C||T is the final cryptography produced.

EAX authentication and encryption steps describe the parallel pipeline structure applied by the CRT encryption mode to process data. OMAC authentication used inside the EAX mode increase the security of the system.

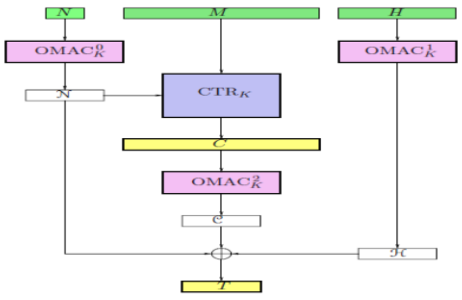


Figure 1: Encryption under EAX. The message is M, the key is K, and the header is H. The ciphertext is CT = Ck||T.

3) *EAX Decryption*

Step 1: if |CT| < r then return INVALID, if size of the Cipher text is less than the tag length, then it get rejected.

Step 2: C||T $\leftarrow$ CT where |T| = r, size of the cipher text is greater than the tag length

Step 3: Step 1: N $\leftarrow$ OMAC^0_K (N), Initial value is Zero, with the OMAC mode nonce is authenticated, N is the result of authentication and this is used as CTR initial vector.

Step 4: H $\leftarrow$ OMAC^1_K (H), Initial value is one, with OMAC mode the header message is authenticated and the H is the result of this process.

Step 5: e $\leftarrow$ OMAC^2_K (C), initial value is two, with OMAC mode the cryptography C is authenticated and the e is the result of this process.

Step 6: Tag\`$\leftarrow$ N (XOR) e (XOR) H, Integrated authentication sign tag is created in this process.

Step 7: T $\leftarrow$ Tag\` |first r bits|, final authentication sign T is produced in this process.

Step 8: if T≠T\`, then return INVALID,

Step 9: M $\leftarrow$ CTR^N_K (C), Initial value is N, with CTR mode message is authenticated and the M is the cryptography after the decryption.

Step 10: return M, M is the plaintext got at end of decryption process. This algorithm is implemented in the android chat application to provide a better security for the message transmission and to the data.

III. CTR AND OMAC

A. *Counter encryption mode(CTR)*

CTR is also known as counter mode; it provides a simplest way to encrypt data using the block cipher. Counter mode is fully parallelizable to make it more efficient in various contemporary usage methods than mode as like CBC [12].

To encrypt data using CTR encryption mode user need to start with plaintext M, key K, counter ctr and ctr is known as n-bit string. Let's consider c be the XOR of M at the same time first |M| bits from the pad of E^k(ctr)||E^k(ctr + 1)||E^k(ctr + 2)…. So, ciphertext C will be generated using the plaintext and ctr. Therefore, decryption of plaintext from the encrypted data is also same like the above process with M and C is interchanged to generate plaintext.

B. One Key Message Authentication Code (OMAC)

OMAC is known as block cipher-based message authentication code which is designed for providing security for the message. OMAC stands for One-Key CBC MAC. Generally it takes only one key, K bits of a block cipher E to perform the message authentication process in the system. It is highly optimized comparing with other message authentication algorithms at the same it is highly efficient as like CBC MAC [14]. It provides a security for arbitrary length message. Anyhow, key length k bits are very low because the underlying block cipher contains k-bit key K. OMAC is general name for the OMAC1 and OMAC2.

C. Architecture of Message Authentication Code (MAC)

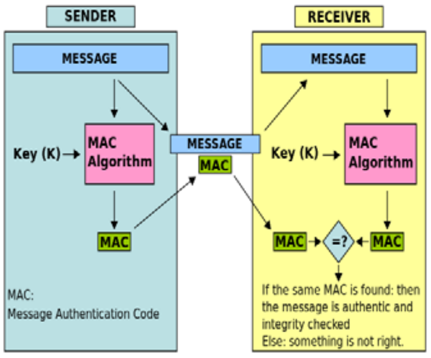

SENDER
RECEIVER
MESSAGE
MESSAGE
Key (K)
MAC Algorithm
MESSAGE
MAC
Key (K)
MAC Algorithm
MAC
MAC
=?
MAC
MAC:
Message Authentication Code
If the same MAC is found: then the message is authentic and integrity checked
Else: something is not right.

Figure 2: Architecture of MAC

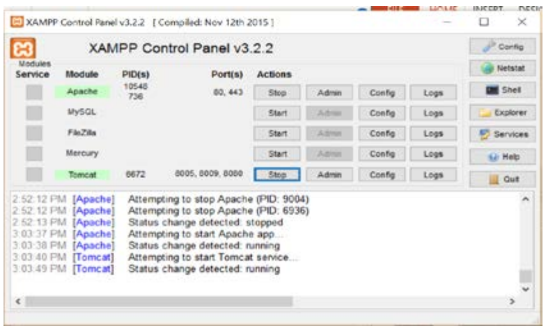
Firstly, message sender needs to runs the data through MAC algorithm to generate MAC data tag. The message including with the MAC tag is transmitted to the receiver of the message. Once, receiver got these details then user need to run the decryption process using the same MAC algorithm with same key. It helps to the message receiver to generate second MAC data tag [11]. The message receiver can differentiate both MAC tag which receiver got on the transmission and Mac tag generated using decryption. Based on the difference between the MAC tag receiver can identify if the message is accessed by someone else in the internet and the message is modified or not.(see Figure 2.)

IV. Implementation of Android Chat Application

This project was implemented on a Lenovo laptop running a windows 7 OS [24] with a 2.40GHz quad core processor, 8 GB of RAM and Intel i5 processor [25]. Eclipse LUNA 1.4.0 [26] integrated development environment (IDE) was used in implementing this project using the Java [27] programming language and the Android 2.2 [28] API technology for demonstrating the application in real-time. A control panel, XAMPP [29] is enclosed to create a local web server for the testing purposes between MySQL 5.5 database [30] and the TOMCAT 7.0 server [31] and SQLYog 2.2 [32], a GUI tool for the MySQL database was used to manage the database(user details).

A. Chatting application database

Android chatting application use MySQL database as a backend to store application user's personal details such as full name, email id and password to login into the system. We can easily implement a communication link between android chatting application and to the MYSQL. Xampp control panel in Figure 3 is used to monitor and to create database in the system. We need to start this control panel to create database and for the other process. Once this control panel starts to process then we can easily access the MYSQL in http://localhost/.


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Config
Netstat
Shell
Explorer
Services
Help
Quit

Modules Service	Module	PID(s)	Port(s)	Actions			
	Apache	10548 736	80, 443	Stop	Admin	Config	Logs
	MySQL			Start	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat	6672	8005, 8009, 8080	Stop	Admin	Config	Logs

```
2:52:12 PM  [Apache]   Attempting to stop Apache (PID: 9004)
2:52:12 PM  [Apache]   Attempting to stop Apache (PID: 6936)
2:52:13 PM  [Apache]   Status change detected: stopped
3:03:37 PM  [Apache]   Attempting to start Apache app...
3:03:38 PM  [Apache]   Status change detected: running
3:03:40 PM  [Tomcat]   Attempting to start Tomcat service...
3:03:49 PM  [Tomcat]   Status change detected: running
```

Figure 3: local server with the help of XAMPP

B. Architechure of Chatting application with EAX mode authentication

The chatting application uses simple protocols to encrypt and transmit messages from one mobile to another mobile. The below figure 4 provides detailed instruction about the steps involved in the proposed android application and the use EAX algorithm in the system. Firstly, users need to register with the system by providing personal details such as username, email id and password to authenticate in the system. These details will be stored in the application data base for the future use, at the same time user's password will be encrypted using the EAX encryption process and stored in the database. Once the registration process is completed then every user of the system will get individual user name and password to login into the system. User of the system can use either their email id or user name to login into the android application. Once user successfully logged into the system then they can access the system functionalist. They can view other users who all are online in the system so they can send or receive message with the help of the application [11]. User can select any available user in the system to send message which is encrypted by the EAX encryption algorithm and a random key is generated by the system which is changed every time while sending and receiving message. This key generation process will start automatically in the system when users start to send message to the other in the application.

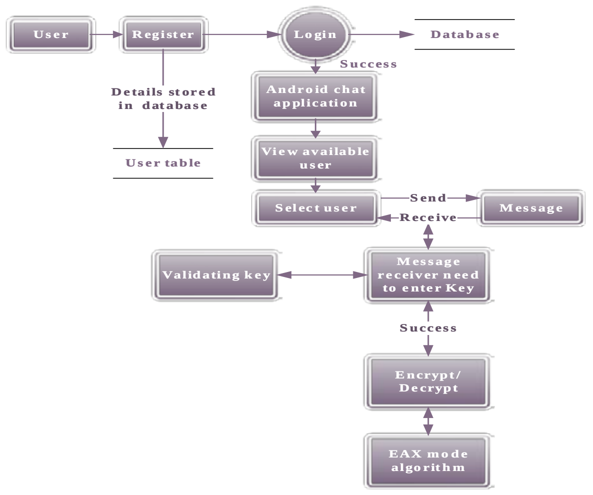
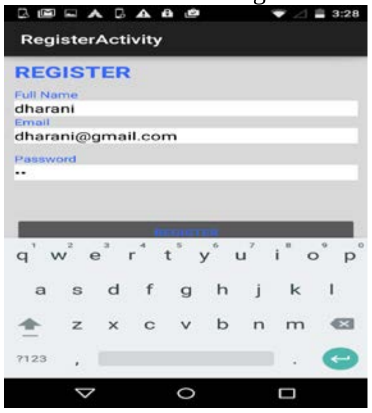


Figure 4: Android chatting application architecture.

Once user sent the message to particular user in the system then the key will be automatically transmitted to the corresponding receiver through any one medium such as text, call or email. At the same time message receiver will get a pop-up window which asks for the key. Message receiver need to enter the key and then the system will validate the key with the encrypted data. If the key will match with the encrypted data then it invokes the decryption process to convert the encrypted text into a plaintext. Only one time message receiver need to authenticate to get access permission to encrypted data.

1) Chatting application user registration

In android chatting application, user needs to register their details to create their own account by entering their full name, email-id and password as shown in Figure 5.


RegisterActivity

REGISTER

Full Name: dharani

Email: dharani@gmail.com

Password: ••

Figure 5: Registration screen of android application

2) Storing user details in database

After user enters their details in the registration page of the application these details will be stored in the database (see Figure 6). This data base contains all details about the registered user.

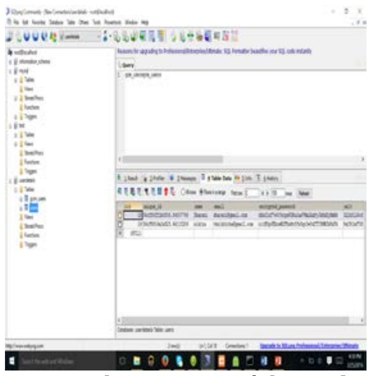
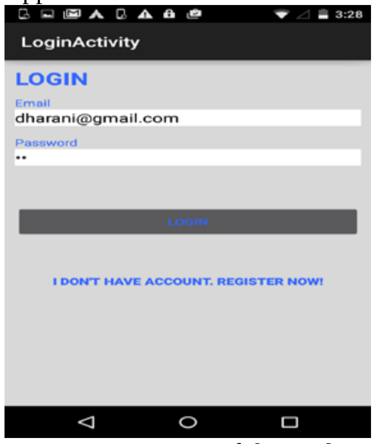


Figure 6: Database screen of the application

3) Login in to the android mobile application

Figure 7 is the login page of the android chatting application, where user need to enter their email-id and password which given during registration process to get logged in to the application.


LoginActivity

LOGIN

Email: dharani@gmail.com

Password: ••

LOGIN

I DON'T HAVE ACCOUNT. REGISTER NOW!

Figure 7: Login screen of the application

4) Chatting application users list

After user logged into their account in the application, they can able to see the list of user, so they can select particular user for sending secret message to them (see figure 8).

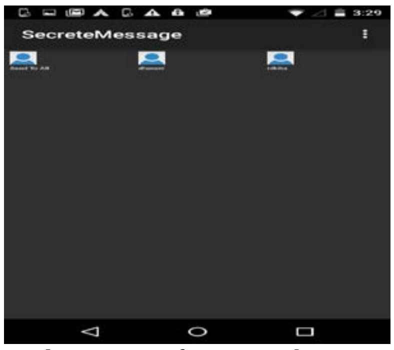
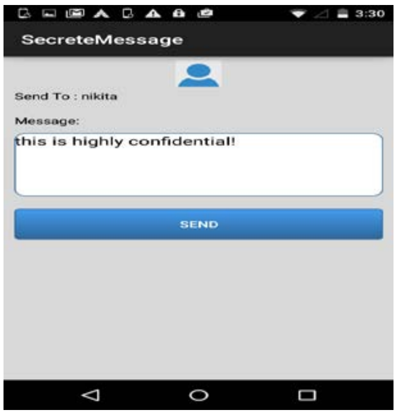


Figure 8: Selecting user for secret chat in application

5) Mobile 1Message sent to the mobile 2

In this application page user can write their message which they want to send encrypted to another user after the user selects the receiver as shown in Figure 9.



Send To : nikita

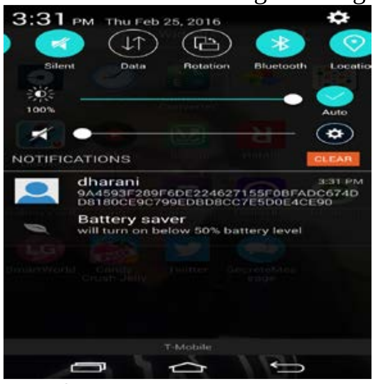
Message:

this is highly confidential!

Figure 9: Message which needs to be encrypted in the application.

6) Mobile2 Notification of the encrypted message from mobile 1

Figure 10 shows the notification of the message in encrypted manner in the receivers mobile. Then the OTP key will be send to the receiver user through message.



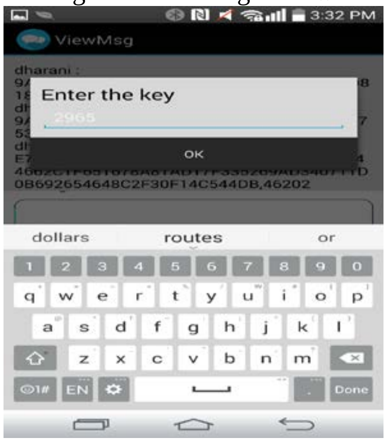
dharani 3:31 PM
9A4593F289F6DE224627155F0BFADC674DD8180CE9C799EDBD8CC7E5D0E4CE90

Battery saver
will turn on below 50% battery level

Figure 10: Notification alert to the receiver with key to retrieve original message.

7) Decryption of the plain text using key at Mobile 2

To view the original message, user in the receiver side of the application need to enter the OTP key which is send to them through message as seen in Figure 11..



dharani :

Enter the key

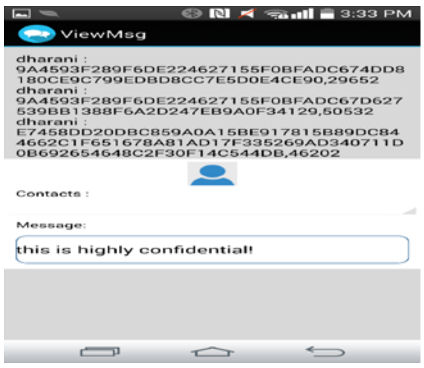
2965

0B692654648C2F30F14C544DB,46202

Figure 11: Popup window to enter key.

8) Successfully encrypted message at mobile 2

After entering the OTP key by the user in the receiver side, the application will decrypt the encrypted message and shows the original message to the user as in Figure 12.



dharani :
9A4593F289F6DE224627155F0BFADC674DD8180CE9C799EDBD8CC7E5D0E4CE90,29652
dharani :
9A4593F289F6DE224627155F0BFADC67D627539BB1388F6A2D247EB9A0F34129,50532
dharani :
E7458DD20DBC859A0A15BE917815B89DC844662C1F651678A81AD17F335269AD340711D0B692654648C2F30F14C544DB,46202

Contacts :

Message:

this is highly confidential!

Figure 12: Decrypted message.

C. *Hardware and Software requirements*

This application was implemented using java programming language which is interlinked to a local server TOMCAT and all the details of the users and their messages are stored in the local database host XAMPP and the data is sent from server to users' devices, and receive messages from devices on the same connection via GCM (Google Cloud Messaging) in windows PC. Java provides a room to the android application developers. Server is used in this application to store user's personal details and also to transmit the encrypted messages to the receiver.

D. *Functionalities of Chat application*

This section offer detailed information about various steps involved in the chatting application.

Firstly, users of this android chatting application need to register to get access permission. Once they completed the above process in the application then they will get user name and password to authenticate in the application. In this registration process application will collect user personal details and store in the data base for the future use.

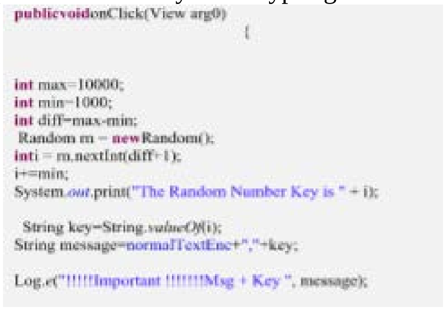
Secondly, users need to provide authentication credential to get access permission form the application. Once user entered their user name and password in the application, then it need match with the information that is stored in the database. If login credential match with the database information then only the application allow user to get access to it functionalities else it generate error message.

Once, user entered in the application then he/she can select any available user for the secret chat. The message will be automatically encrypted using EAX algorithm in the application background and transmitted to the receiver. Message receiver need to enter “key” to invoke decryption program in the application. The decryption process also done as same as encryption process of the application.

V. Performance Analysis and Sample Coding

A. *Source Code*

In figure11, a random number is generated which is represented as a key in encrypting the message



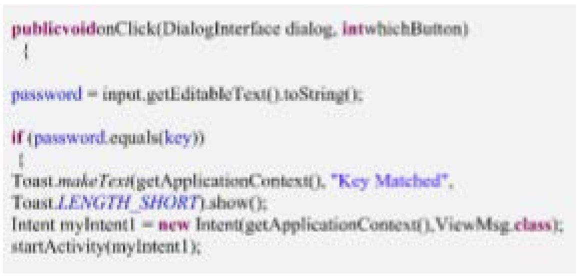
```
publicvoidonClick(View arg0)
                                        {


int max=10000;
int min=1000;
int diff=max-min;
 Random rn = newRandom();
inti = rn.nextInt(diff+1);
i+=min;
System.out.print("The Random Number Key is " + i);

  String key=String.valueOf(i);
String message=normalTextEnc+","+key;

Log.e("!!!!!Important !!!!!!Msg + Key ", message);
```

Figure 11: Random number Generator



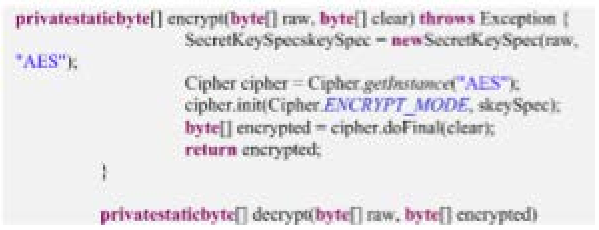
```
publicvoidonClick(DialogInterface dialog, intwhichButton)
  {

password = input.getEditableText().toString();

if (password.equals(key))
  {
Toast.makeText(getApplicationContext(), "Key Matched",
Toast.LENGTH_SHORT).show();
Intent myIntent1 = new Intent(getApplicationContext(),ViewMsg.class);
startActivity(myIntent1);
```

Figure 12: Random Key Activator

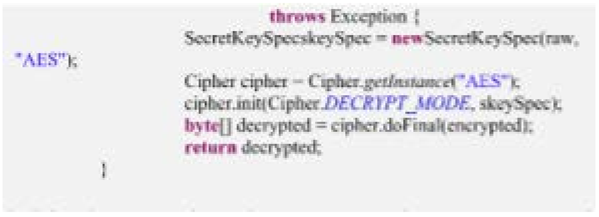
Here, it checks if the key entered is matching the generated one or not



```
privatestaticbyte[] encrypt(byte[] raw, byte[] clear) throws Exception {
                        SecretKeySpecskeySpec = newSecretKeySpec(raw,
"AES");
                        Cipher cipher = Cipher.getInstance("AES");
                        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
                        byte[] encrypted = cipher.doFinal(clear);
                        return encrypted;
                }

                privatestaticbyte[] decrypt(byte[] raw, byte[] encrypted)
```

Figure 13:Encryption



```
                              throws Exception {
                        SecretKeySpecskeySpec = newSecretKeySpec(raw,
"AES");
                        Cipher cipher = Cipher.getInstance("AES");
                        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
                        byte[] decrypted = cipher.doFinal(encrypted);
                        return decrypted;
                }
```

Figure 14: Decryption

In figure 13 & 14, the AES algorithm is implemented in order to encrypt and decrypt the message

B. *Performance Analysis*

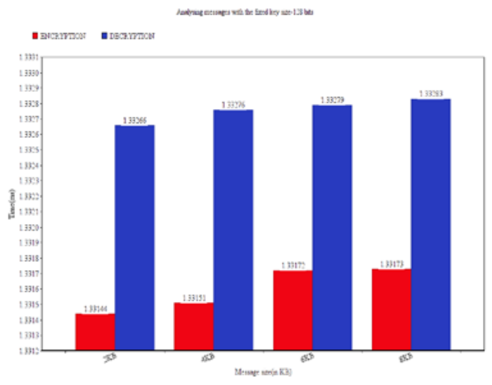


Figure 15: Performance analysis for variable message length with a fixed key size

In figure 15, we can observe how much time it has taken to encrypt and decrypt the messages of various message lengths by maintaining the key size to 128 bits. This made us analyze that it does take only a few nanoseconds to encrypt and decrypt any message. Next, as shown in figure 16, we kept the message length fixed (1KB) and altered the key sizes to 128,192 and 256 bits to analyze the performance. It is shown that increase in key sizes, increases the encryption and decryption times.

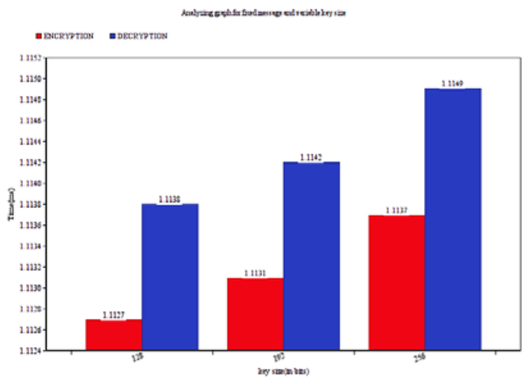


Figure 16: Performance analysis for variable key sizes with a fixed message length

VI. Conclusion and Future Work

This EAX mode of algorithm creates a new way of communication and share information securely with others without any issues. It helps to increase the confidence of internet users to share their confidential information. The proposed system is designed for providing security for the messages. By modifying the process and functionalities in the EAX mode of algorithm, we can provide security for the videos and audio files to share via the network. Currently, this android chat application is designed to deal with the messages to transmit in the network with high security. In the future this application may be broaden to include some additional futures such as file transfer, video chat and voice chat in the secured way.

VII. References

[1] ´E. Jaulmes, A. J. (Springer-Verlag, 2002). On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption, FSE 2002, LNCS 2365*, 237–251.

[2] IEEE Standard for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables. (2012). *IEEE 1703-2012*.

[3] American National Standard Protocol Specification For Interfacing to Data Communication Networks. (2008). *ANSI C12.22-2008.*

[4] Housley R, W. D. (2008). CCM: AES Mode of Operation. *http: //csrc.nist.gov/encryption/modes/proposedmodes/.*

[5] Iwata, K. K. (2003). TMAC: Two-Key CBC MAC. *Cryptology ePrint Archive, Report 2002/092, http://eprint.iacr.org/. To appear in Cryptographers Track RSA Conference 2003*.

[6] Iwata, T. K. (2003). OMAC: One-Key CBC MAC. *In: Johansson, T. (ed.) FSE. Lecture Notes in Computer Science, vol. 2887*, 129–153.

[7] Iwata, T. K. (2003). Stronger Security Bounds for OMAC, TMAC, and XCBC. *In: Johansson, T., Maitra, S. (eds.) INDOCRYPT. Lecture Notes in Computer Science, vol. 2904*, 402–415.

[8] Joan Daemen, V. a. (2008). AES Proposal. *Rijndael, http://www.mathmagic.cn/bbs/ftp/ AES%20Proposal%20Rijndael.pdf.*

[9] Kurosawa, T. I. (2007). OMAC: One-Key CBC MAC. *http://crypt.cis.ibaraki.ac.jp/omac/omac.pdf.*

[10] Kurosawa, T. I. (March 10, 2003). OMAC: One-Key CBC MAC. *Department of Computer and Information Sciences,Ibaraki University*.

[11] Kurosawa., T. I. (2003). OMAC: One-key CBC MAC. *Fast Software Encryption '03, Lecture Notes in Computer Science Vol. ?? , T. Johansson ed., Springer-Verlag, 2003. Also Cryptology ePrint archive Report 2002/180, http://eprint.iacr.org/2002/180*.

[12] Liskov, M. R. (2011). Tweakable block ciphers. *J. Cryptology 24(3)*, 588–613.

[13] M. Bellare, P. R. (January 18 2004). The EAX Mode of Operation,A Two-Pass Authenticated-Encryption Scheme Optimized forSimplicity and Efficiency. *http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf.*

[14] Measurement Canada. (2009). Specification for Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol to Complement the Utility Industry End Device Data Tables.

[15] Minematsu, K. L. (2012). Cryptanalysis of EAX-Prime. *DIAC - Directions in Authenticated Ciphers*.

[16] Moise, A. B. (May 2011). EAX' Cipher Mode . *NIST Submission (2011), http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax-prime/eax-prime-spec.pdf.*

[17] P. Rogaway, T. S. (August 20, 2007). The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption. *http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/siv/siv.pdf.*

[18] Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. (n.d.). *NIST SP 800-38B http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.*

[19] Rogaway, J. B. (n.d.). Comments to NIST concerning AES modes of operations:A suggestion for handling arbitrary-length messages with the CBC MAC. *Second Modes of Operation Workshop. Available at http://www.cs.ucdavis.edu/~rogaway/.*

[20] Rogaway, J. B. (Springer-Verlag, 2000.). CBC MACs for arbitrary-length messages: The three key constructions. *Advances in Cryptology — CRYPTO 2000, LNCS 1880*, 197–215.

[21] Rogaway, J. B. (Springer-Verlag, 2002). A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — EUROCRYPT 2002, LNCS 2332*, 384–397.

[22] Rogaway, M. B. (Springer-Verlag, 2006). The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. *Advances in Cryptology – EUROCRYPT '06, Lecture Notes in Computer Science Vol. , ed.*

[23] Tackmann, M. B. (July 2013). Insecurity of MtE (and M&E) AEAD. *Personal communications (unpublished note).*

[24] Microsoft Windows Corporation. *http://windows.microsoft.com/en-us/windows7/products/upgrade#T1=tab01*.

[25] Intel Corporation. *http://www.intel.com/content/www/us/en/processors/core/core-i5-processor.html*

[26] Eclipse. *https://projects.eclipse.org/projects/soa.jwt/releases/1.4.0/review*

[27] Oracle Corporation, Java *https://www.oracle.com/java/index.html*

[28] GoogleCorporation, *http://developer.android.com/about/versions/android-2.2.html*

[29] Apache XAMPP, *https://www.apachefriends.org/index.html*

[30] Oracle Corporation, MySQL. *http://www.mysql.com/*

[31] Apache, *https://tomcat.apache.org/download-70.cgi*

[32] WebYog,SQLYog.*https://www.webyog.com/#, http://sqlyogkb.webyog.com/article/166-what-is-sqlyog*