ANDROID-BASED ENCRYPTED SMS SYSTEM- SafeChatApp

Computer Networks and Security Project
Submitted to Prof. Ramesh Babu Battula

YUKTI KHURANA
2017UCP1234

Malaviya National Institute Of Technology, Jaipur

INTRODUCTION

The name of this Android Application is SafeChatApp. It is based on an Advanced Encryption and Decryption (AES) Android System or App that provides messaging service for Android users. The users having an account can send messages to any other user to begin a chatting session with them. The messages sent by the user are encrypted using user's unique current session "Private Key". Unless the receiver enters the sender's private key, they cannot read the decrypted legible message. To provide further security, the user can use a distinct encryption key on each login. The app acts like any normal chatting application but with added encryption security to prevent eavesdropping.

WHY DO WE NEED ENCRYPTION?

- In this world of socialising from even miles of distance, we usually rely on chatting apps and SMS services to keep in touch with our loved ones.
- Messages that are not encrypted, meaning the contents of each text message are view-able to mobile carriers and governments, can even be intercepted by organized and semi-skilled hackers or precisely malicious - crackers. That means even after securing your online accounts using two-factor authentication, your codes can be stolen.
- Just as bad, such messages can leak metadata, which is information about the message but not the contents of the message itself, such as the phone number of the sender and the recipient, which can identify the people involved in the conversation.

WHY DO WE NEED ENCRYPTION?

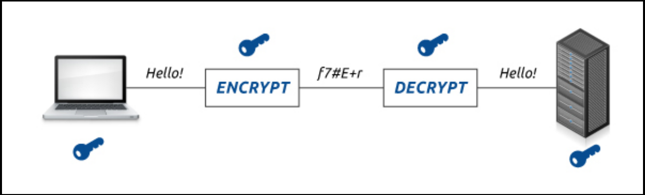
- There have been many amazing platforms to quench our thirst of socialising with our friends and family. But are they safe to use?
- Raw messages can also be spoofed, meaning you can never be completely sure that a message came from a particular person.
- How can we be sure that our messages are not being seen by a third party or attacker or malicious entity. Even when our messages might not have top secret military information, nobody wants their private conversations to be read by someone else. So what is the alternative?

Do we stop interacting? NO

The answer is ENCRYPTION !!

WHAT IS ENCRYPTION?

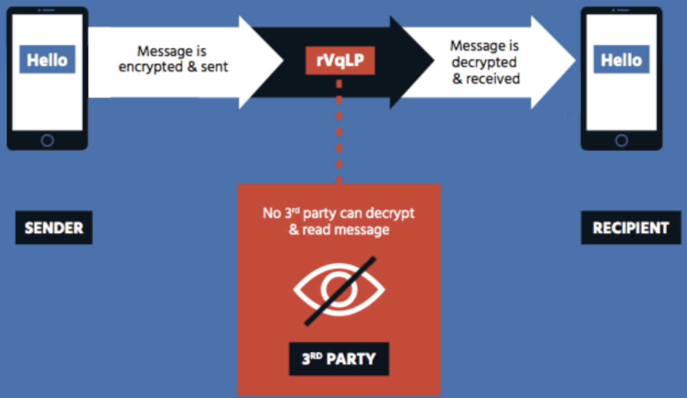
- Encryption can be defined as the process of converting information or data into a code, especially to prevent unauthorized access.
- Encryption helps jumble the content of a message into random data using a key based on some encryption algorithmthat the hackers cannot decrypt without the private key.
- The message stays in cryptic form until it's received on the other end and the original message is compiled back together again. This means if anyone intercepts the message – a hacker for example – it's mostly just jumbled characters and symbols.



WHAT IS ENCRYPTED MESSAGING?

- If you want to send messages without worrying that other people might be poking around in the texts you're sending, you should be using an encrypted messaging service..
- Encrypted messaging (also known as secure messaging) provides end-to-end encryption for user-to-user text messaging. Encrypted messaging prevents anyone from monitoring your text conversations.
- Many encrypted messaging apps also offer end-to-end encryption for phone calls made using the apps, as well as for files that are sent using the apps.
- Examples of some encrypted messaging apps are WhatsApp, Viber, Line, Telegram, Cyphr, KakaoTalk, Silence, and many more.

END-TO-END ENCRYPTION


Hello
Message is encrypted & sent
rVqLP
Message is decrypted & received
Hello
SENDER
No 3rd party can decrypt & read message
3RD PARTY
RECIPIENT

WHAT IS END-TO-END ENCRYPTION

- End-to-end (E2E) encryption is any form of encryption in which only the sender and intended recipient hold the keys to decrypt the message. The most important aspect of E2E encryption is that no third party, even the party providing the communication service, has knowledge of the encryption keys.
- It can solve the problem of revealing data while net sniffing if a Web server has been compromised. If implemented with trusted algorithms, end-to-end encryption can provide the highest level of data protection.

INTRODUCTION TO MY SAFECHAT APP

- I have made an android-app called SafeChatApp through which users can easily interact with one another in a secure manner.
- The app is based on Advanced Encryption Algorithm that encrypts the messages of the sender using the key provided by them.
- These encrypted messages are decrypted at the receiver's end only if the receiver enters the same private key. This ensure that no third party not even the communication provider can eavesdrop their conversation.

THE SECURE CHATTING APP

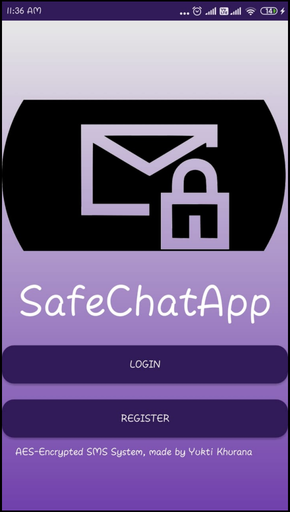


Figure: START PAGE

THE SECURE CHATTING APP

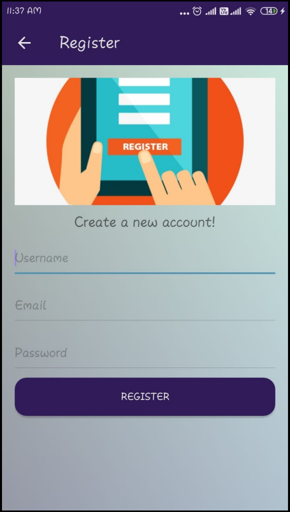
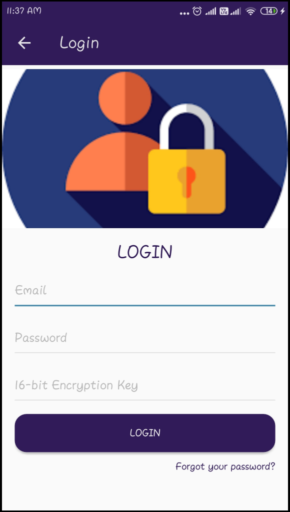


Figure: REGISTER PAGE

THE SECURE CHATTING APP


11:37 AM

Login

LOGIN

Email

Password

16-bit Encryption Key

LOGIN

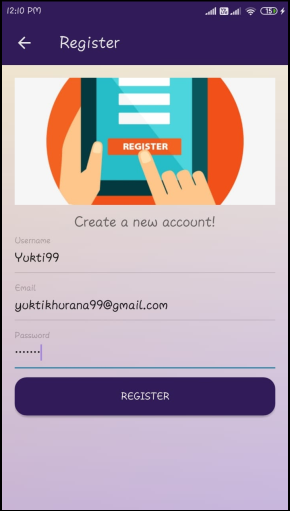
Forgot your password?

Figure: LOGIN PAGE

HOW TO USE THE APP ?

- The app opens up with a start-up page offering a choice between LOGIN or REGISTER.
- To use the app, the user is required to first create an account using their email, a password with at least 6 characters and a username of their choice.
- After successful registration, the user can login to the main activity page of the chat application.
- For Login, the user must enter a 16-character encryption key to encrypt messages sent in the current logged in session. This key will act as the shared private key between the two interacting parties.

Create an Account on MySafeChat App


12:10 PM

Register

Create a new account!

Username
Yukti99

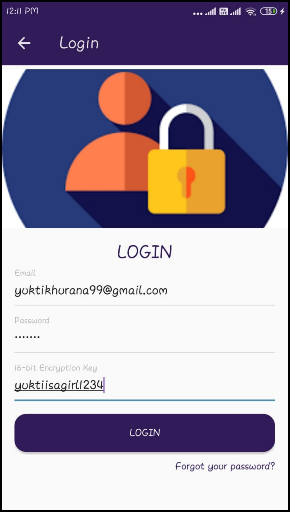
Email
yuktikhurana99@gmail.com

Password
•••••••

REGISTER

Figure: 1. Create your account by entering your email, a password and a username

Login to your new Account


12:11 PM

Login

LOGIN

Email
yuktikhurana99@gmail.com

Password
•••••••

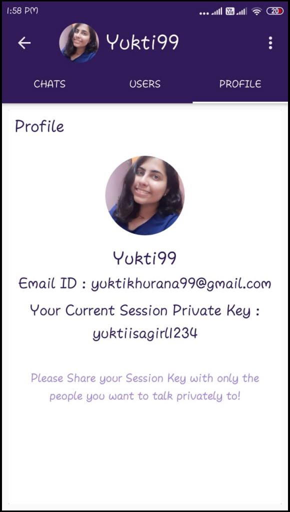
16-bit Encryption Key
yuktiisagirl1234

LOGIN

Forgot your password?

Figure: 2. Login to your account and enter the 16-character session key

Home Page of your Account


1:58 PM

Profile

Yukti99

Email ID : yuktikhurana99@gmail.com

Your Current Session Private Key : yuktiisagirl1234

Please Share your Session Key with only the people you want to talk privately to!

Figure: 3. Profile Tab shows info about your profile, your username, session key, email and profile picture. You can change your username and pic in this

BEGIN SECURE CHATTING

- After successful login into the application, users may search from any users under the “USERS” tab of the home screen layout.
- You can now normally text any of your friends or acquaintances.
- The only catch is that as soon as you hit the send button your messages shall be encrypted using your current session private key that you entered during your login.
- This will provide security to your chatting experience.

AT SENDER'S END

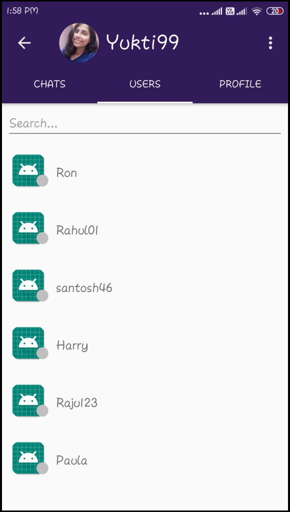
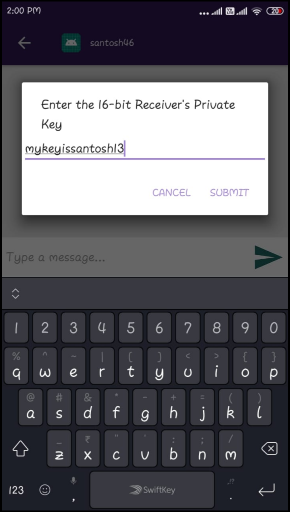


Figure: 4.Users Tab, showing a list of all the users of the application

AT SENDER'S END


2:00 PM

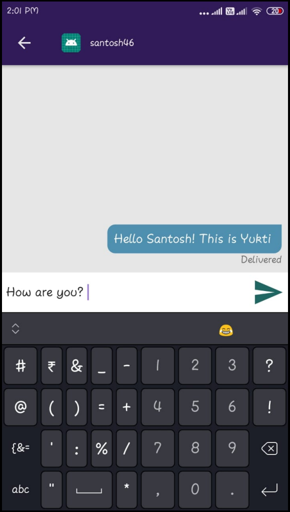
santosh46

Enter the 16-bit Receiver's Private Key

mykeyissantosh13

Figure: 5.To chat with Santosh, enter her encryption Key

AT SENDER'S END


2:01 PM

santosh46

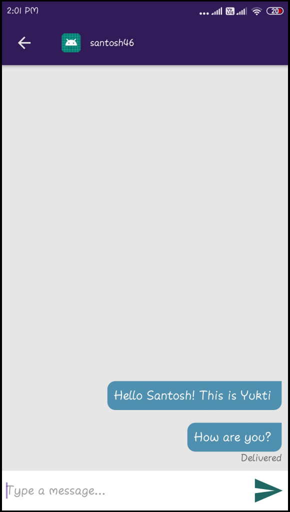
Hello Santosh! This is Yukti

Delivered

How are you?

Figure: 6.Once you send your message it is encrypted by your encryption key, in the background database, not visible to sender

AT SENDER'S END


2:01 PM

santosh46

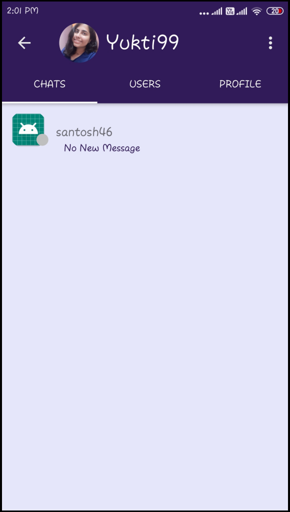
Hello Santosh! This is Yukti

How are you?

Delivered

Figure: 7.The messages sent by Yukti are visible to herself for ease. It is also noted if the receiver has "SEEN" them yet or not

AT SENDER'S END



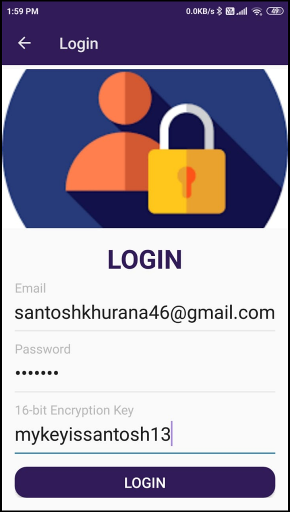
santosh46
No New Message

Figure: 8.Santosh added to your recent chat tab, so conversation can continue easily.

AT RECEIVER'S END

- As soon as the receiver logins into their account, they receive the notification of a message from a user.
- When the receiver clicks the chat to view the messages, they are prompted for the 16-character encryption key of the sender.
- Only if the user enters the correct key, can he read the decrypted message, otherwise, the message will seem just illegible gibberish.
- This will provide further authorisation and privacy to both entities. On top of it, the messages sent by the receiver this time, will be encrypted using their distinct session key. So the sender must be aware of receiver's key to read their messages.

AT RECEIVER'S END


1:59 PM 0.0KB/s 49

Login

LOGIN

Email
santoshkhurana46@gmail.com

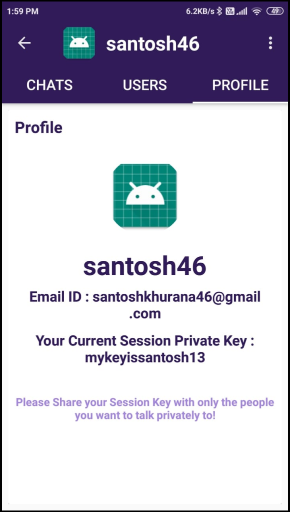
Password
•••••••

16-bit Encryption Key
mykeyissantosh13

LOGIN

Figure: 9.Santosh logs in using her account email and password, along with a session key

AT RECEIVER'S END



Profile

santosh46

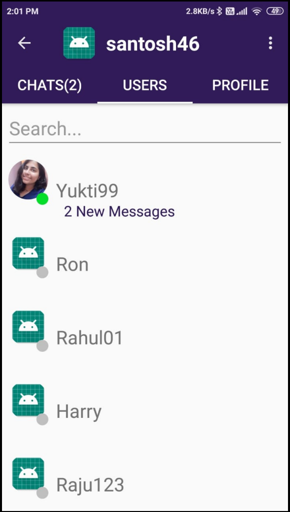
Email ID : santoshkhurana46@gmail .com

Your Current Session Private Key : mykeyissantosh13

Please Share your Session Key with only the people you want to talk privately to!

Figure: 10.Santosh's Profile Information

AT RECEIVER'S END


2:01 PM

2.8KB/s

santosh46

CHATS(2) USERS PROFILE

Search...

Yukti99
2 New Messages

Ron

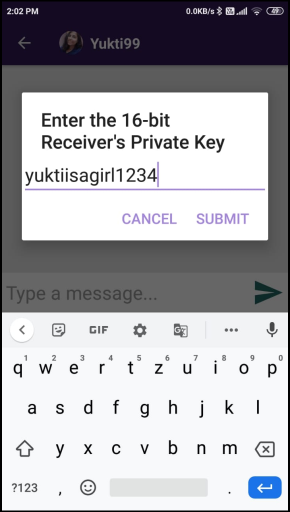
Rahul01

Harry

Raju123

Figure: 11.The messages received by Yukti are shown to Santosh on her home page

AT RECEIVER'S END


2:02 PM

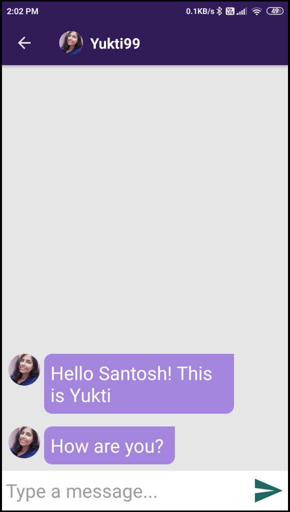
Yukti99

Enter the 16-bit Receiver's Private Key

yuktiisagirl1234

Figure: 12.To read the message sent by Yukti, Santosh (the receiver) must enter the encryption key of Yukti

AT RECEIVER'S END

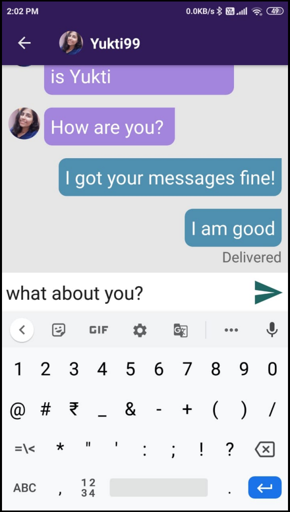


Hello Santosh! This is Yukti

How are you?

Figure: 13.Now, Santosh can read Yukti's messages!

SECURE CONVERSATIONS USING AES


2:02 PM

Yukti99

is Yukti

How are you?

I got your messages fine!

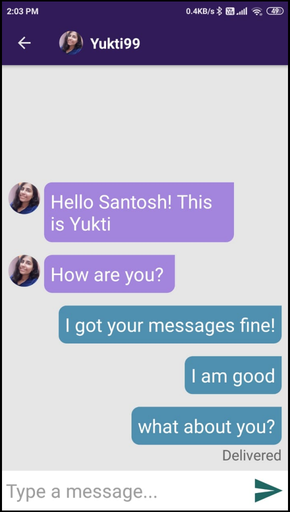
I am good

Delivered

what about you?

Figure: 14.Now, Santosh can continue the conversation in a similar way

SECURE CONVERSATIONS USING AES


2:03 PM

Yukti99

Hello Santosh! This is Yukti

How are you?

I got your messages fine!

I am good

what about you?

Delivered

Figure: 15.But the messages sent by Santosh will be encrypted using her session key.

SECURE CONVERSATIONS USING AES

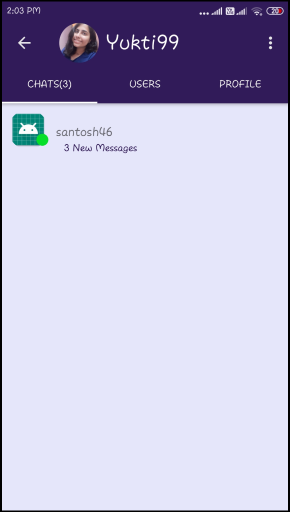
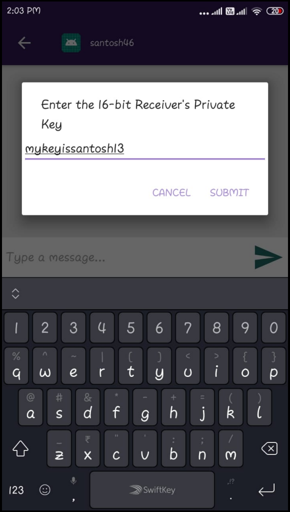


Figure: 16.Yukti will receive notification of Santosh's messages and shall use Santosh's key to read/decrypt them

SECURE CONVERSATIONS USING AES



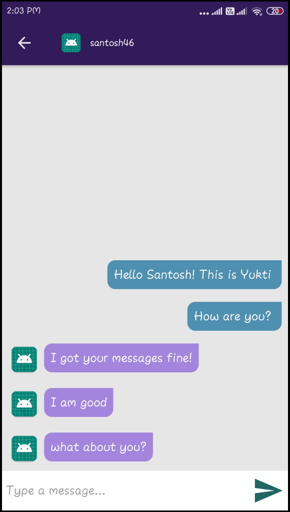
Enter the 16-bit Receiver's Private Key

mykeyissantosh13

CANCEL SUBMIT

Figure: 17.Yukti will type Santosh's encryption key

SECURE CONVERSATIONS USING AES



Hello Santosh! This is Yukti

How are you?

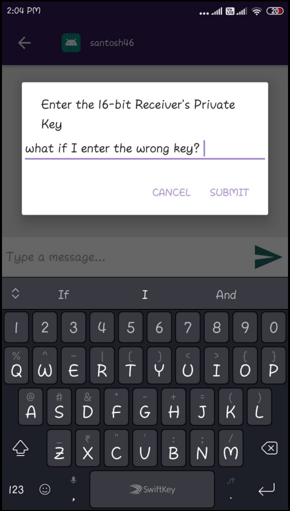
I got your messages fine!

I am good

what about you?

Figure: 18.Now, Yukti can read Santosh's messages easily and securely!

WHAT IF THE WRONG KEY IS ENTERED?


2:04 PM

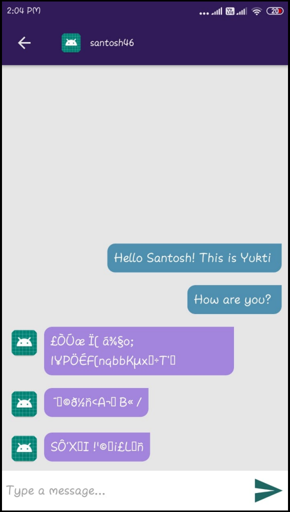
santosh46

Enter the 16-bit Receiver's Private Key

what if I enter the wrong key?

Figure: 19.Suppose the receiver enters a wrong key

WHAT IF THE WRONG KEY IS ENTERED?



Hello Santosh! This is Yukti

How are you?

£ÒŬœ Ï(ā¾§o;
I¥PÖÉF(nqbbKµx□÷T'□

¯□©ð½ñ<A¬□ B« /

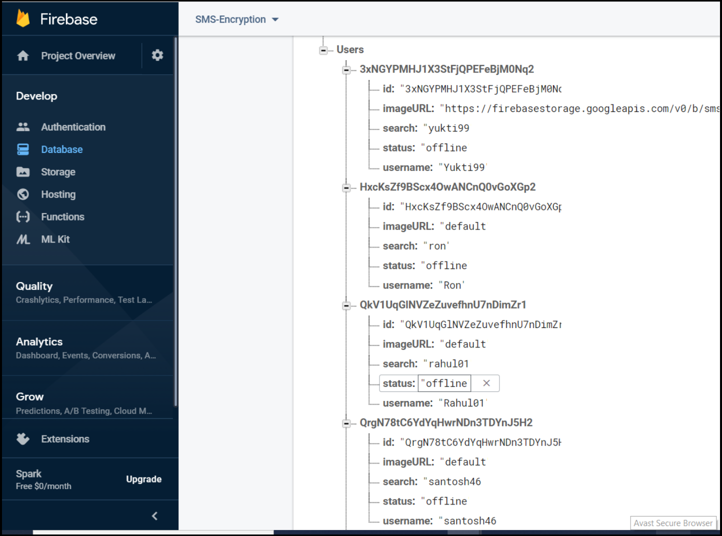
SÔ'X□I !'©□i£L□ñ

Figure: 20.The receiver won't be able to read the received messages, it will seem like all gibberish.

BACK-END FIREBASE DATABASE

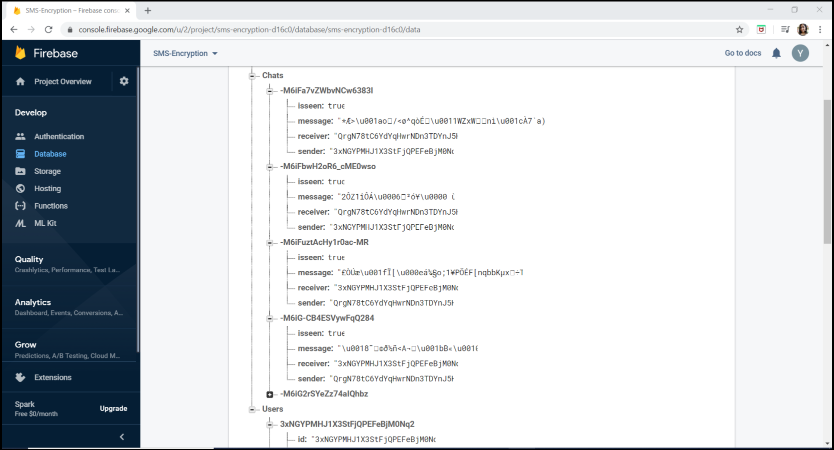
- Firebase online database has been used to store information about the app.
- The users table as shown created in the firebase whenever a user creates an account on SafeChatApp. providing the communication service, has knowledge of the encryption keys.
- The user information stored is: id, username, image url, status and search.

BACK-END FIREBASE DATABASE



```
Users
  3xNGYPMHJ1X3StFjQPEFeBjM0Nq2
    id: "3xNGYPMHJ1X3StFjQPEFeBjM0Nc
    imageURL: "https://firebasestorage.googleapis.com/v0/b/sms
    search: "yukti99
    status: "offline
    username: "Yukti99'
  HxcKsZf9BScx4OwANCnQ0vGoXGp2
    id: "HxcKsZf9BScx4OwANCnQ0vGoXGp
    imageURL: "default
    search: "ron'
    status: "offline
    username: "Ron'
  QkV1UqGlNVZeZuvefhnU7nDimZr1
    id: "QkV1UqGlNVZeZuvefhnU7nDimZr
    imageURL: "default
    search: "rahul01
    status: "offline
    username: "Rahul01'
  QrgN78tC6YdYqHwrNDn3TDYnJ5H2
    id: "QrgN78tC6YdYqHwrNDn3TDYnJ5H
    imageURL: "default
    search: "santosh46
    status: "offline
    username: "santosh46
```

CHATS STORED IN ENCRYPTED FORM


SMS-Encryption – Firebase conso
console.firebase.google.com/u/2/project/sms-encryption-d16c0/database/sms-encryption-d16c0/data
Firebase
SMS-Encryption
Go to docs
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test La...
Analytics
Dashboard, Events, Conversions, A...
Grow
Predictions, A/B Testing, Cloud M...
Extensions
Spark
Free $0/month
Upgrade

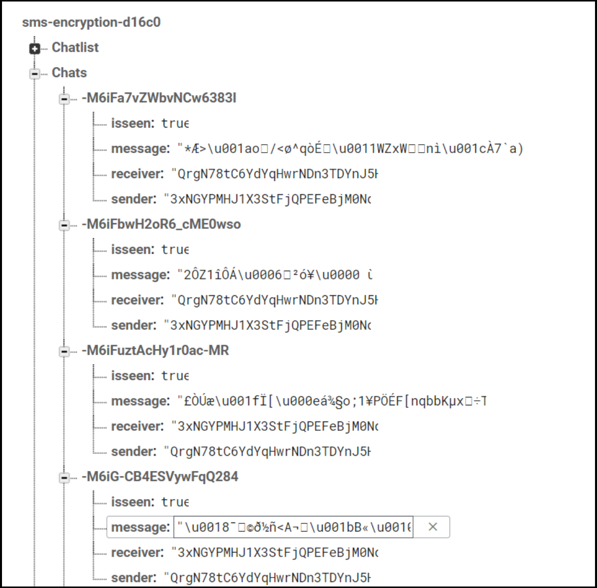
```
Chats
  -M6iFa7vZWbvNCw6383I
    isseen: true
    message: "*Æ>\u001ao□/<ø^qòÉ□\u0011WZxW□□nì\u001cÀ7`a)
    receiver: "QrgN78tC6YdYqHwrNDn3TDYnJ5
    sender: "3xNGYPMHJ1X3StFjQPEFeBjM0N
  -M6iFbwH2oR6_cME0wso
    isseen: true
    message: "2ÔZ1îÔÁ\u0006□²ó¥\u0000 ì
    receiver: "QrgN78tC6YdYqHwrNDn3TDYnJ5
    sender: "3xNGYPMHJ1X3StFjQPEFeBjM0N
  -M6iFuztAcHy1r0ac-MR
    isseen: true
    message: "£ÒÚæ\u001fÏ[\u000eá¾§o;1¥PÖÉF[nqbbKµx□÷T
    receiver: "3xNGYPMHJ1X3StFjQPEFeBjM0N
    sender: "QrgN78tC6YdYqHwrNDn3TDYnJ5
  -M6iG-CB4ESVywFqQ284
    isseen: true
    message: "\u0018¯□©ð½ñ<A¬□\u001bB«\u001
    receiver: "3xNGYPMHJ1X3StFjQPEFeBjM0N
    sender: "QrgN78tC6YdYqHwrNDn3TDYnJ5
  -M6iG2rSYeZz74aIQhbz
Users
  3xNGYPMHJ1X3StFjQPEFeBjM0Nq2
    id: "3xNGYPMHJ1X3StFjQPEFeBjM0N
```

Figure: The chats are stored in encrypted for, so even the communication provider cannot eavesdrop

ENCRYPTED CHATS

- The chats stored in Chats Table of the SMS-encryption database, stores all the information in encrypted form..
- The sender and receiver's id is stored, to reload previous conversations at a later time.
- The message is stored in encrypted form so that there is complete privacy between sender and receiver and even the communication provider cannot read/ eavesdrop the conversation.

ENCRYPTED CHATS



```
sms-encryption-d16c0
 ⊞ Chatlist
 ⊟ Chats
    ⊟ -M6iFa7vZWbvNCw6383I
        isseen: true
        message: "*Æ>\u001ao□/<ø^qòÉ□\u0011WZxW□□nì\u001cÀ7`a)
        receiver: "QrgN78tC6YdYqHwrNDn3TDYnJ5
        sender: "3xNGYPMHJ1X3StFjQPEFeBjM0N
    ⊟ -M6iFbwH2oR6_cME0wso
        isseen: true
        message: "2ÔZ1îÔÁ\u0006□²ó¥\u0000 ì
        receiver: "QrgN78tC6YdYqHwrNDn3TDYnJ5
        sender: "3xNGYPMHJ1X3StFjQPEFeBjM0N
    ⊟ -M6iFuztAcHy1r0ac-MR
        isseen: true
        message: "£ÒÚæ\u001fÏ[\u000eá¾§o;1¥PÖÉF[nqbbKµx□÷T
        receiver: "3xNGYPMHJ1X3StFjQPEFeBjM0N
        sender: "QrgN78tC6YdYqHwrNDn3TDYnJ5
    ⊟ -M6iG-CB4ESVywFqQ284
        isseen: true
        message: "\u0018¯□©ð½ñ<A¬□\u001bB«\u001
        receiver: "3xNGYPMHJ1X3StFjQPEFeBjM0N
        sender: "QrgN78tC6YdYqHwrNDn3TDYnJ5
```

MORE FEATURES OF THE APP

- The SafeChatApp is very convenient to use and provided complete end-to-end encryption.
- The added security measure of prompting the receiver for encryption key provides authentication, confidentiality as well as integrity.This app ensures that no malicious entity or hacker can steal private conversation information without knowing the key.
- Further security measure employed in this android app is the fact that, each side has its own private key i.e. to read the whole conversation and make sense out of it, the attacker must have the keys of both parties.
- The Key is session-dependent, i.e. in each session the user can choose any key of his/her choice.
- The communication-providing entity i.e. SafeChatApp will have no information about the key. If a user wants, he can change the key for a new session, this will render all the previous conversation unreadable and thus untraceable.

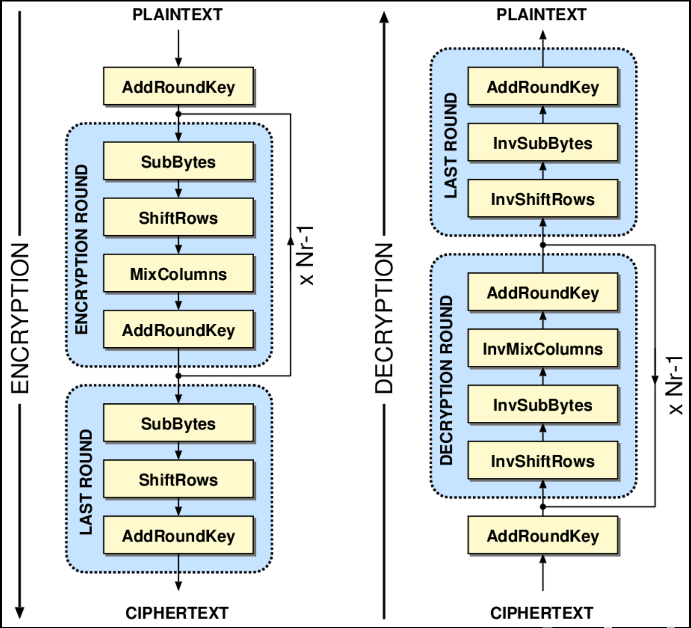
TECHNICAL DETAILS OF THE APP

- NAME: SafeChatApp
- BACKEND: Java
- FRONTEND: XML
- SOFTWARE: Android Studio on Windows 10
- ANDROID STUDIO VERSION: 3.4.2
- DATABASE:Firebase Database & Storage
- MINIMUM SDK VERSION: 16
- TARGET SDK VERSION: 28
- APP PERMISSIONS: Internet Connection, Device's External Storage read/write permissions
- ABOUT: App works perfectly on 84.5% of android devices of matching SDK configurations

AES ENCRYPTION DECRYPTION

- The app uses Advanced Encryption Standard to encrypt and decrypt the messages exchanged between the users.
- AES is one of the most popular and widely adopted symmetric encryption algorithm. It is at least 6 times faster than Triple DES.
- It is a symmetric block cipher that encrypts data in blocks of size 128-bit and uses 128/192/256 keys for encryption.
- The large size of AES keys and multiple rounds of substitution and permutation make it a very strong encryption standard which has not been broken by any known person till now.
- AES-256 with 256 bit key and 14-rounds is the most advance cryptographic algorithm till date and not have been broken yet.
- It is a symmetric block cipher i.e. the same key is used to encrypt and decrypt the plaintext or message. So both sender and receiver must know the same secret key.

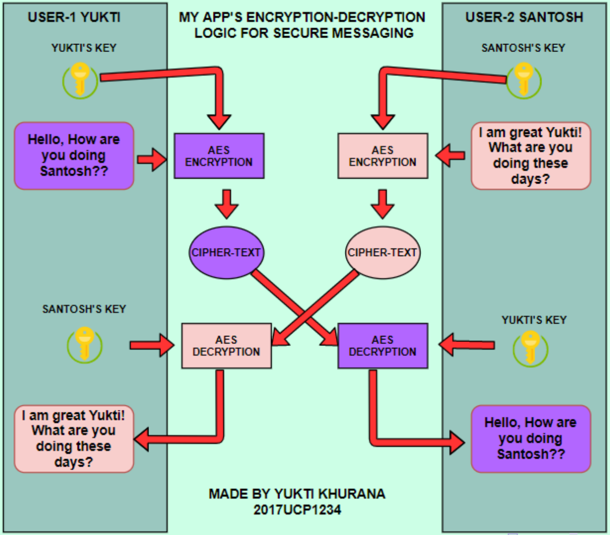
AES ENCRYPTION DECRYPTION



WHY AND HOW AES MAKES THE APP SECURE?

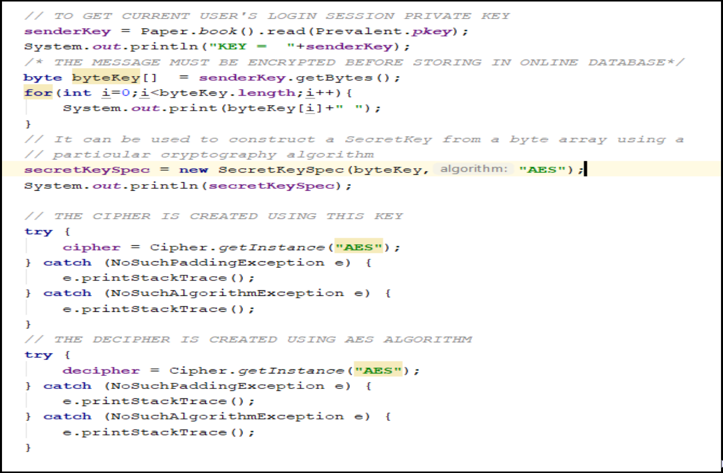
- Its building blocks and design principles are fully specified.
- It has sustained years of attempted cryptanalysis from many smart people, in a high-exposure situation, and it came out relatively unscathed.
- AES makes extensive use of Galois field theory
- AES being a symmetric key algorithm is not known to be reduced to any "known hard problem" which makes it unbreakable for current computational speeds. best key-recovery attacks on full AES so far is a Biclique attack which is merely faster than brute force by at most a factor of 4. Furthermore, the data storage required works out to trillions of terabytes, far beyond the data stored on the planet.
- The strength of AES increases exponentially with Key-size as it leads to more "Confusion"- an important property of any secure cipher.

MY APP'S BACKGROUND ENCRYPTION & DECRYPTION LOGIC


USER-1 YUKTI
MY APP'S ENCRYPTION-DECRYPTION LOGIC FOR SECURE MESSAGING
USER-2 SANTOSH
YUKTI'S KEY
SANTOSH'S KEY
Hello, How are you doing Santosh??
AES ENCRYPTION
AES ENCRYPTION
I am great Yukti! What are you doing these days?
CIPHER-TEXT
CIPHER-TEXT
SANTOSH'S KEY
YUKTI'S KEY
AES DECRYPTION
AES DECRYPTION
I am great Yukti! What are you doing these days?
Hello, How are you doing Santosh??
MADE BY YUKTI KHURANA 2017UCP1234

BACKEND LOGIC FOR AES ENCRYPTION

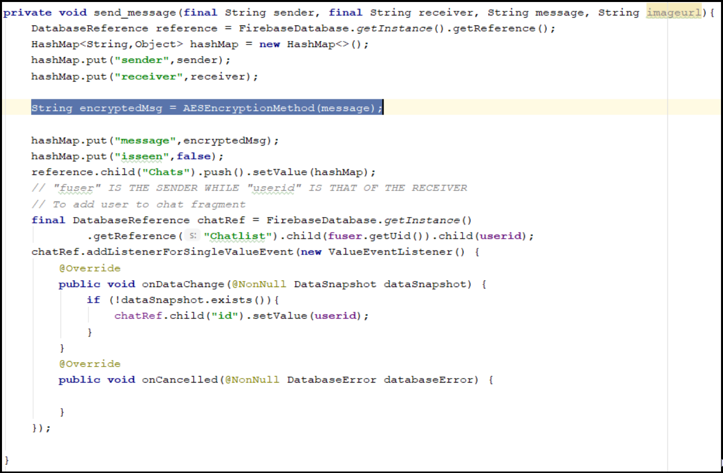
Figure: Java cryptography architecture is used here and the sender's key is the current-session private key used for encryption in the send-message function of MessageActivity Module of the app.



```
// TO GET CURRENT USER'S LOGIN SESSION PRIVATE KEY
senderKey = Paper.book().read(Prevalent.pkey);
System.out.println("KEY =   "+senderKey);
/* THE MESSAGE MUST BE ENCRYPTED BEFORE STORING IN ONLINE DATABASE*/
byte byteKey[]   = senderKey.getBytes();
for(int i=0;i<byteKey.length;i++){
    System.out.print(byteKey[i]+"  ");
}
// It can be used to construct a SecretKey from a byte array using a
// particular cryptography algorithm
secretKeySpec = new SecretKeySpec(byteKey, algorithm: "AES");
System.out.println(secretKeySpec);

// THE CIPHER IS CREATED USING THIS KEY
try {
    cipher = Cipher.getInstance("AES");
} catch (NoSuchPaddingException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
// THE DECIPHER IS CREATED USING AES ALGORITHM
try {
    decipher = Cipher.getInstance("AES");
} catch (NoSuchPaddingException e) {
    e.printStackTrace();
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
}
```

SEND MESSAGE FUNCTION AT SENDER'S END

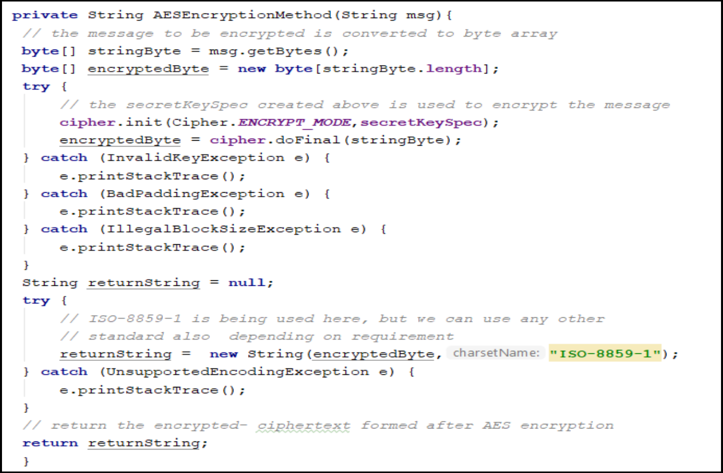
Figure: The send message function of the MessageActivity module encrypts the message using AESEncryptionMethod() function and then stores encrypted data to firebase database instead of the real data.



```
private void send_message(final String sender, final String receiver, String message, String imageurl){
    DatabaseReference reference = FirebaseDatabase.getInstance().getReference();
    HashMap<String,Object> hashMap = new HashMap<>();
    hashMap.put("sender",sender);
    hashMap.put("receiver",receiver);

    String encryptedMsg = AESEncryptionMethod(message);

    hashMap.put("message",encryptedMsg);
    hashMap.put("isseen",false);
    reference.child("Chats").push().setValue(hashMap);
    // "fuser" IS THE SENDER WHILE "userid" IS THAT OF THE RECEIVER
    // To add user to chat fragment
    final DatabaseReference chatRef = FirebaseDatabase.getInstance()
            .getReference( s: "Chatlist").child(fuser.getUid()).child(userid);
    chatRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if (!dataSnapshot.exists()){
                chatRef.child("id").setValue(userid);
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }
    });

}
```

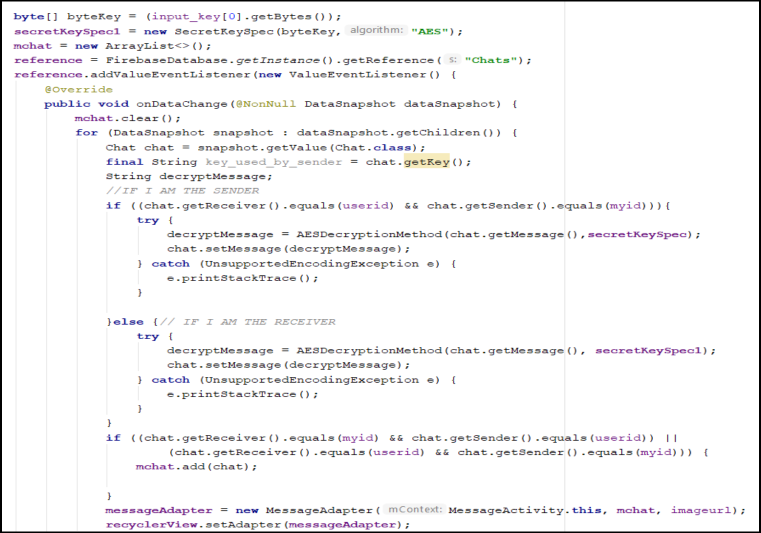
AES ENCRYPTION METHOD AT SENDER'S END

Figure: The message is first converted to a byte array and then encrypt mode and secretKeySpec are created using sender's session key.The standard use here is ISO-8859-1,but any other can also be used.



```
private String AESEncryptionMethod(String msg){
 // the message to be encrypted is converted to byte array
 byte[] stringByte = msg.getBytes();
 byte[] encryptedByte = new byte[stringByte.length];
 try {
     // the secretKeySpec created above is used to encrypt the message
     cipher.init(Cipher.ENCRYPT_MODE,secretKeySpec);
     encryptedByte = cipher.doFinal(stringByte);
 } catch (InvalidKeyException e) {
     e.printStackTrace();
 } catch (BadPaddingException e) {
     e.printStackTrace();
 } catch (IllegalBlockSizeException e) {
     e.printStackTrace();
 }
 String returnString = null;
 try {
     // ISO-8859-1 is being used here, but we can use any other
     // standard also  depending on requirement
     returnString =  new String(encryptedByte, charsetName: "ISO-8859-1");
 } catch (UnsupportedEncodingException e) {
     e.printStackTrace();
 }
 // return the encrypted- ciphertext formed after AES encryption
 return returnString;
 }
```

READ MESSAGE FUNCTION AT RECEIVER'S END



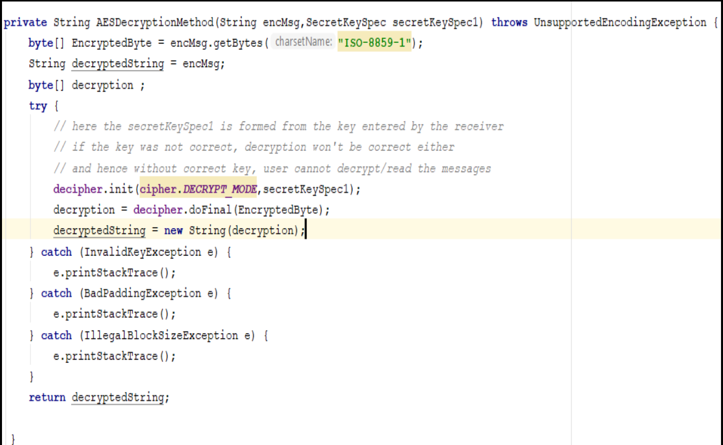
```
byte[] byteKey = (input_key[0].getBytes());
secretKeySpec1 = new SecretKeySpec(byteKey, algorithm: "AES");
mchat = new ArrayList<>();
reference = FirebaseDatabase.getInstance().getReference( s: "Chats");
reference.addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        mchat.clear();
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Chat chat = snapshot.getValue(Chat.class);
            final String key_used_by_sender = chat.getKey();
            String decryptMessage;
            //IF I AM THE SENDER
            if ((chat.getReceiver().equals(userid) && chat.getSender().equals(myid))){
                try {
                    decryptMessage = AESDecryptionMethod(chat.getMessage(),secretKeySpec);
                    chat.setMessage(decryptMessage);
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }

            }else {// IF I AM THE RECEIVER
                try {
                    decryptMessage = AESDecryptionMethod(chat.getMessage(), secretKeySpec1);
                    chat.setMessage(decryptMessage);
                } catch (UnsupportedEncodingException e) {
                    e.printStackTrace();
                }
            }
            if ((chat.getReceiver().equals(myid) && chat.getSender().equals(userid)) ||
                    (chat.getReceiver().equals(userid) && chat.getSender().equals(myid))) {
                mchat.add(chat);

            }
            messageAdapter = new MessageAdapter( mContext: MessageActivity.this, mchat, imageurl);
            recyclerView.setAdapter(messageAdapter);
```

READ MESSAGE FUNCTION AT RECEIVER'S END

- The read message function of the MessageActivity module of app, first prompts the user to enter a 16-character encryption key of the sender.
- The same key is converted to AES key standard and used to decrypt the encrypted message retrieved from the firebase database.
- If the key is correct, the message will be decrypted correctly.For the sender reading his sent messages, the messages are decoded already for him to read easily.

AES DECRYPTION METHOD AT RECEIVER'S END

Figure: This function returns the decrypted version of the encrypted message given, using the key provided by the receiver.If the key is entered correctly by the receiver, the data will be decrypted correctly and user will be able to read it.



```
private String AESDecryptionMethod(String encMsg,SecretKeySpec secretKeySpec1) throws UnsupportedEncodingException {
    byte[] EncryptedByte = encMsg.getBytes( charsetName: "ISO-8859-1");
    String decryptedString = encMsg;
    byte[] decryption ;
    try {
        // here the secretKeySpec1 is formed from the key entered by the receiver
        // if the key was not correct, decryption won't be correct either
        // and hence without correct key, user cannot decrypt/read the messages
        decipher.init(cipher.DECRYPT_MODE,secretKeySpec1);
        decryption = decipher.doFinal(EncryptedByte);
        decryptedString = new String(decryption);
    } catch (InvalidKeyException e) {
        e.printStackTrace();
    } catch (BadPaddingException e) {
        e.printStackTrace();
    } catch (IllegalBlockSizeException e) {
        e.printStackTrace();
    }
    return decryptedString;

}
```

CONCLUSION

- Given the exceptional strength of AES Algorithm, This android app will ensure a secure chatting experience for users who don't want their conversations to be eavesdropped by any third entity.
- Every user session is end-to-end encrypted in such a manner that there is not inconvenience to the user.
- The users communicating must enter their session-encryption keys that both of them must be aware of and they can start messaging!

THANK YOU

CYBER-SECURITY/ COMPUTER NETWORKS AND SECURITY PROJECT
MADE BY YUKTI KHURANA
2017UCP1234
6TH SEMESTER, 3RD YEAR BTECH
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR