SMS Encryption using AES Algorithm on Android

Rohan Rayarikar
B.E in Computer Engineering

Sanket Upadhyay
B.E in Computer Engineering

Priyanka Pimpale
B.E in Computer Engineering

ABSTRACT
Encryption is of prime importance when confidential data is transmitted over the network. Varied encryption algorithms like AES, DES, RC4 and others are available for the same. The most widely accepted algorithm is AES algorithm. We have developed an application on Android platform which allows the user to encrypt the messages before it is transmitted over the network. We have used the Advanced Encryption Standards algorithm for encryption and decryption of the data. This application can run on any device which works on Android platform. This application provides a secure, fast, and strong encryption of the data. There is a huge amount of confusion and diffusion of the data during encryption which makes it very difficult for an attacker to interpret the encryption pattern and the plain text form of the encrypted data. The messages encrypted by the developed application are also resistant to Brute-Force and pattern attacks. The various uses of this application in real life and its functionality are explained in this paper.

General Terms
Security Algorithm, Symmetric Key Encryption, Android Application, SMS.

Keywords
SMS, AES, Android, Application.

1. INTRODUCTION
The application developed for end to end secure transmission of the SMS. The algorithm used is Advanced Encryption Standards algorithm. This application is developed on Android platform and is one of a kind. The later part of the paper explains the working of SMS, the AES algorithm and the working of our developed application.

1.1 Need for secure data transmission
Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction.

Maintaining privacy in our personal communication is something everyone desires.

Encryption is a means to achieve that privacy. It was invented for the very same purpose. [5] As short message service (SMS) is now widely used as a business tool; its security has become a major concern for business organization and customers. There is a need for an end to end SMS encryption in order to provide a secure medium for communication.

1.2 Literature Survey
Recent trends in enterprise mobility have made mobile device security an imperative. IDC reported in 2010 that for the first time smartphone sales outpaced PC sales. Faced by this onslaught of devices and recognizing the productivity and cost benefits, organizations are increasingly implementing bring-your-own device (BYOD) policies. Research firm J. Gold Associates reports that about 25%-35% of enterprises currently have a BYOD policy, and they expect that to grow to over 50% over the next two years. This makes sense as mobility evolves from a nice-to-have capability to a business advantage.

But the competitive edge and other benefits of mobility can be lost if smartphones and tablet PCs are not adequately protected against mobile device security threats. While the market shows no sign of slowing, IT organizations identify security as one of their greatest concerns about extending mobility. Therefore, various encryption techniques are used. [2]

Encryption has long been used by militaries and governments to facilitate secret communication. Encryption is now commonly used in protecting information within many kinds of civilian systems. For example, the Computer Security Institute reported that in 2007, 71% of companies surveyed utilized encryption for some of their data in transit, and 53% utilized encryption for some of their data in storage [3]

Encryption can be used to protect data "at rest", such as files on computers and storage devices (e.g. USB flash drives). In recent years there have been numerous reports of confidential data such as customers' personal records being exposed through loss or theft of laptops or backup drives. Encrypting such files at rest helps protect them should physical security measures fail. [2] Digital rights management systems which prevent unauthorized use or reproduction of copyrighted material and protect software against reverse engineering (see also copy protection) are another somewhat different example of using encryption on data at rest.

In 2010, 6.1 trillion SMS text messages were sent. This translates into 192,192 SMS per second. SMS has become a massive commercial industry, worth over $81 billion globally as of 2006. The global average price for an SMS message is $0.11, while mobile networks charge each other interconnect fees of at least $0.04 when connecting between different phone networks.

The SMS industry being on such a great rise is vulnerable to attacks. Therefore it has now become more imperative to encrypt SMS before sending.[3]

Various algorithms for encryption and decryption are in place. Out of the entire group of algorithm AES is the most preferred one.

AES require very low RAM space and its very fast. On Pentium Pro processors AES encryption requires only 18 clock cycles/byte equivalent to throughput of about 11Mib/s for 200MHz processor. This was the main reason why we decided to use AES algorithm for encryption and decryption. [6]

There are few SMS applications on Google Play which encrypts the SMS using AES algorithm.

We have programmed our application meticulously considering various factors which might benefit the user. With only less than 200Kb size it is feather weight which effectively makes it faster. It provides functionality like conversation view, Inbox, Draft, Backup and restore; all the functionality which a standard SMS application should provide. The main advantage is that it is very simple app, easy to understand and very easy to operate. User interface is so simple and light weight that main functionality of encryption

and decryption of SMS is carried out very efficiently.

2. SHORT MESSAGE SERVICE (SMS)

SMS stands for **short message service**. Simply put, it is a method of communication that sends text between cell phones, or from a PC or handheld to a cell phone. The "short" part refers to the maximum size of the text messages: 160 characters (letters, numbers or symbols in the Latin alphabet). For other alphabets, such as Chinese, the maximum SMS size is 70 characters.

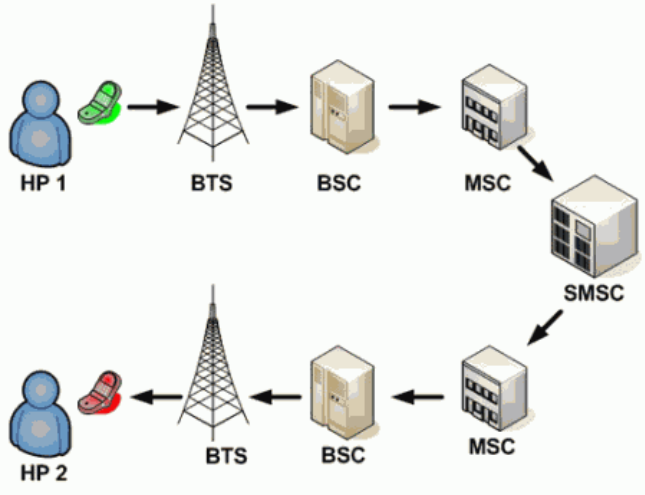

Fig. 1: Transmission of SMS

2.1 Working of SMS

It is well-known that SMS service is a cell phone feature but indeed, SMS can also work on other computing devices such as PC, Laptop, or Tablet PC as long as they can accept SIM Card. SIM Card is needed because SMS service needs SMS center client which is built-in on the SIM Card.

2.1.1 BTS

A base transceiver station (BTS) is a piece of equipment that facilitates wireless communication between user equipment (UE) and a network. UEs are devices like mobile phones (handsets), WLL phones, computers with wireless internet connectivity, WiFi and WiMAX devices and others.

2.1.2 MSC

The mobile switching center (MSC) is the primary service delivery node for GSM/CDMA, responsible for routing voice calls and SMS as well as other services (such as conference calls, FAX and circuit switched data).[2]

The MSC sets up and releases the end-to-end connection, handles mobility and hand-over requirements during the call and takes care of charging and real time pre-paid account monitoring.

2.1.3 SMSC

When SMS is transmitted from a cell phone, the message will be received by mobile carrier's SMS Center (SMSC), do destination finding, and then send it to destination devices (cellphone).

SMSC is SMS service center which is installed on mobile carrier core networks. Beside as SMS forwarding, SMSC also acts as temporary storage for SMS messages. So, if the destination cell phone is not active, SMS will store the message and then deliver it after the destination cell phone is active. As additional, SMSC also notify the sender whether the SMS delivering is success or not. However SMSC cannot store the SMS message forever since the storage capacity is not unlimited.

During the SMS delivering, sender cell phone and SMSC is actively communicating.

So, if the non-active destination cell phones become active, SMSC directly notifies the sender cell phone and tell that the SMS delivering is success.

This is how the SMS works in general. The following part describes the AES algorithm.

3. ADVANCE ENCRYPTION STANDARDS ALGORITHM/ RIJNDAEL ALGORITHM

The Advanced Encryption Standard comprises three block ciphers, AES-128, AES-192 and AES-256. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits. The block-size has a maximum of 256 bits, but the key-size has no theoretical maximum. The cipher uses number of encryption rounds which converts plain text to cipher text. The output of each round is the input to the next round. The output of the final round is the encrypted plain text known as cipher text. The input given by the user is entered in a matrix known as State Matrix. [2]

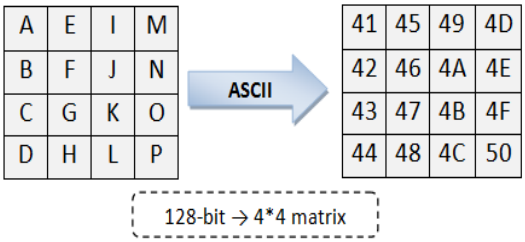

Fig. 2: State Matrix

Following are the four steps.

3.1 SubBytes Step

This step is same as SubBytes step of AES algorithm. In the S-Box Substitution step, each byte in the matrix is reorganized using an 8-bit substitution box. This substitution box is called the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF (28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points. [7] This step causes confusion of data in the matrix. S-Box Substitution is carried out separately for LPT and RPT. This is the first step of iterative round transformation. The output of this round is given to the next round. [3]

3.2 ShiftRows Step

The ShiftRows step is performed on the rows of the state matrix. It cyclically shifts the bytes in each row by a certain offset. The first row remains unchanged. Each byte of the second row is shifted one position to the left. Similarly, the third and fourth rows are shifted by two positions and three positions respectively. The shifting pattern for block of size 128 bits and 192 bits is the same.[3]

3.3MixColumns Step

In the MixColumns step, the four bytes of each column of the state matrix are combined using an invertible linear transformation [5]. A randomly generated polynomial is arranged in a 4*4 matrix. The same polynomial is used during decryption. Each column of the state matrix is XOR-ed with

the corresponding column of the polynomial matrix. The result is updated in the same column. The output matrix is the input to AddRoundKey.[3]

3.4 AddRoundKey

A round key is generated by performing various operations on the cipher key.

This round key is XOR-ed with each byte of the state matrix. For every round a new round key is generated using Rijndael's key scheduling algorithm. [3]

3.5 Decryption of the Proposed Algorithm

The encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher. In addition, the cipher and the inverse cipher operations must be executed in such a way that they cancel each other. The rounds keys must also be used in reverse order. [4] The Cipher Text which is formed of 256-bit 4*8 Matrix is the input for the decryption process. [3]

3.6 Implementation

The algorithm can be implemented in any language. This algorithm can also be used in Image Processing. We have implemented it in java, java being an open source and platform independent language. The pseudo codes for the components of the cipher are given below. [3]

3.6.1 Add Round key:

```
public byte[ ][ ]  addRoundKey(byte[ ][ ] state,byte[ ][ ]
roundkey)
{
      for (int i=0;i<4;i++)
     {
         for (int j=0;j<4;j++)
            {
            state [i][j]=doExclusiveOR(state[i][j],
roundkey[i][j]);
            }
     }
     return state;
}
```

3.6.2 Substitute Bytes:

```
public byte[ ][ ] subBytes(byte[][] state)
{
      for (int i=0;i<4;i++)
      {
            for (int j=0;j<4;j++)
            {
                        int row = getFirstFourBits(state[i][j]);
                        int column =
getSecondFourBit(state[i][j]);
                        state[i][j] =
sBoxSubstitution(row,column);
            }
      }
      return state;
}
```

3.6.3 MixColumns:

```
public byte[ ][ ] mixColumns(byte[ ][ ] state)
{
    for (int c=0;c<4;c++)
    {
        state [c]=matrixMultiplication(state[c], polynomial);
    }
    return state;
}
```

3.6.4 ShiftRows:

```
shiftRows(byte state[][])
{
for(int i=0;i<4;i++)
{
//cyclic left shifts 'i'th row, 'i'times
cyclicLeftShift(i);
}
}
```

3.7 Strength of the Algorithm

The cipher key used in the algorithm is of 128 bits. Therefore, to break the cipher key an attacker has to check 2^{128} possibilities which are practically almost impossible. Therefore, the **Brute-force Attack** fails on this algorithm.

The flow of the algorithm makes sure that there is no fixed pattern in any of the steps of the algorithm. The components of the proposed algorithm have brought about strong diffusion and confusion. Therefore, **statistical and pattern analysis** of the ciphertext fails. [4]

The most important security advantage is that no **differential** or **linear attacks** can break this algorithm. [9]

4. SMS APPLICATION

The application works in following way:

1. The user opens the application and authenticates using pattern lock.
2. User can either type new message or reply to an existing message.
3. If new message is selected, user enters the message and presses encrypt button after inserting the recipient's name. The user has to enter a cipher key before the message is sent. The cipher key is auto-generated if the user does not enter one.
4. If the user selects to reply to an existing message, he first decrypts the message by long pressing the message and then types in the reply. The user is asked to enter cipher key before the message is sent.
5. Once the cipher key is entered, the message is successfully sent and is shown in encrypted form in the thread.

4.1 Application Snapshots

Some of the snapshots of the application are shown below. It should be noted that due to obvious reasons we are not sharing the entire layout of the application. However, few of the important snapshots are given below.

4.1.1 Pattern Lock

Fig. 3: Pattern Lock

This is used by the user to authenticate his identity. The user may change the lock code once he authenticates and logs into the application. After 5 incorrect attempts the application closes.

4.1.2 Create Message

The user types in the message along with the name of the sender. The relevant contact information is displayed in the dropdown menu as the user starts to write the name. He can then select the name and the number which is displayed in dropdown menu. If suppose user types 'ro' in the name field then all the contacts having initials 'ro' or containing 'ro' as a sub-text are displayed in the dropdown menu below it along with the telephone number.

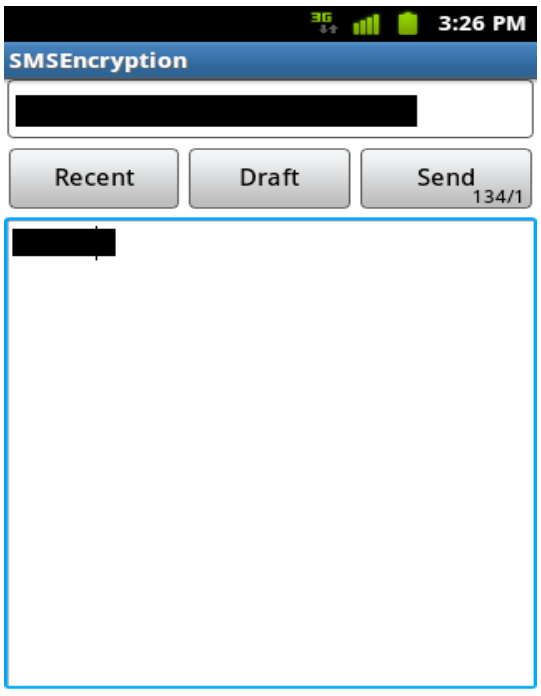

Fig. 4: Create Message

User may save the message as draft or send it by entering the cipher key. User can choose any recent contact from his call log by pressing 'Recent' button.

4.1.3 Thread View

The messages in the application inbox are shown in form of thread. Long pressing on the thread gives option to delete the thread or open the contact information of the thread or call the contact to whom the thread belongs

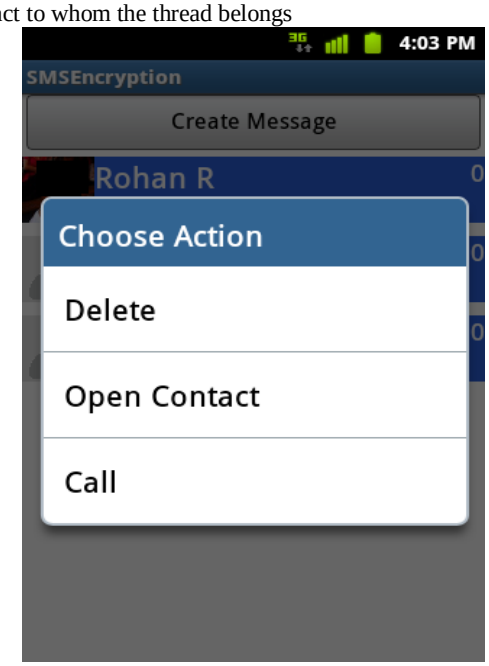

Fig. 5: Message Inbox

4.1.4 Thread View

The messages are displayed in thread format. For ease of understanding we have shown the first two messages in the encrypted form and the last two are in decrypted form. The encrypted message is decrypted by using AES decryption.

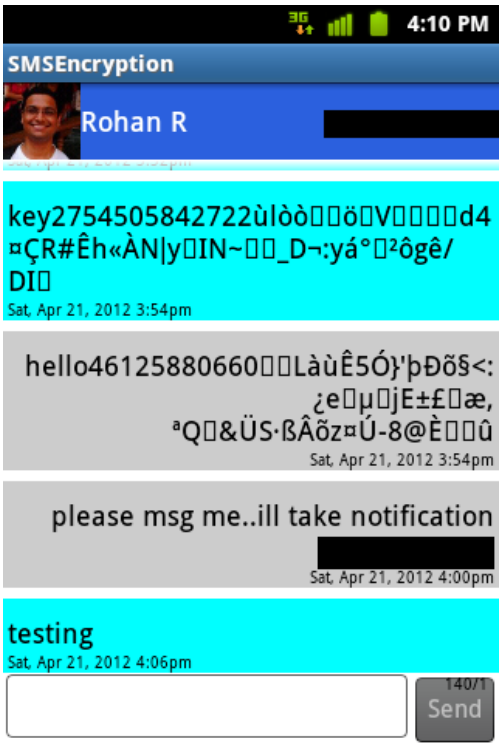

Fig. 6: Messages in Encrypted and Decrypted Form

4.2Features of this Application

Following are some of the features of the application:

1. All messages in thread are displayed in encrypted format to both sender and receiver.
2. Long pressing the thread will pop-up an action box wherein the user can delete, view contact details or call the recipient.
3. Long pressing any message in the thread will pop-up an action box wherein the user can delete, forward or decrypt the message.
4. The cipher key is randomly generated if the user does not enter it.
5. Various settings such as notification settings, Display settings, Encryption settings, Tone settings, Personalization settings are available for the user's convenience.
6. This application is developed on Android platform. The reason behind using Android platform is similar to other operating systems for mobile devices; Android OS supports connectivity, messaging, language support, media support, Bluetooth etc. The main feature of android would be open source technology and JAVA support. It also supports multitasking, multi touch, Wi-Fi, tethering, 3G services, and very importantly security and privacy.

4.3 Goals of this application

The main goals of our application are:

1. Developing a secure SMS application.
2. Maintaining encrypted information of message recipients.
3. Decrypting of message as per users requirement.
4. Protection against misuse of message information.
5. High confidentiality and improved security

4.3.1 Commercial Domain

In some commercial setups it is very must that information flow between various departments remain private and other department people should not come to know. This application can be used in such cases where the numbers and digits have more importance than documentations. The proposed application can be used for secured transactions on network. [5]

4.3.2 Non-Commercial and Personal Use

There are sometimes when the user would like to keep talks between two people private and confidential. During such times, SMS encryption is a boon. An intruder would not be able to understand the message unless he has a proper authentication key.

4.4 Scope

The application is built on Android platform. Therefore, it can be used on any device which runs on Android operating system. This application can be used in industries for secured data transfer. Apart from commercial and business use, this application can be used for non-commercial and personal use. [6] The purpose of this application is secured data transfer between two devices.

4.5 Pseudo Codes of Android Application

We have written the code in Android language. The original codes are not given for obvious reasons. However, the main logic of the codes are given.

4.5.1 Send Message

In Android, There is a class SmsManager. We create instance of this class and there is a sendTextMessage() method in SmsManager class.

```
void sendSMS(String text,String number)
{
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(number,null,text,null,null);

// last two parameters in sendTextMessage method are
PendingIntent
// sentIntent & deliveryIntent.
}
```

4.5.2 Receive Message

For receiving any messages we create one BroadcastReciver. And we override onRecieve method of it which is basically called by system when any messages are received. But to do so we first have to register our reciver.

```
public void onRecive(Context c, Intent i)
{
Bundle b=i.getExtras();
if(b!=null)
{
// Retrive received message
byte[] pdus=bundle.get("pdus");

// converting bytes into Message
SmsMessages[] msgs=new SmsMessages[pdus.length];

for(int j=0;j<msgs.length;j++)
{
msgs[j]=SmsMessage.createFromPdu(pdus[j]);
// Here we can display the Sms by getDisplayMessageBody()
method of SmsMessage class
}
}
}
```

4.5.3 Notification

Using NotificationManager and Notification classes we can easily create and display notifications on receiving message.

```
public void createNotification(Context ctx)
{
NotificationManager notifManager =
(NotificationManager)getSystemService(NOTIFICATION_S
ERVICE);

// create object of Notification class
Notification notification = new Notification();

// set the notification details. Here last parameter is
PendingIntent
notification.setLatestEventInfo(ctx, "title", "notification text",
null);

// Notify the system
notifManager.notify(0,notification);

}
```

5. CONCLUSION

As a conclusion the requirements for speed and compactness were met. The program size is 50 kB and it can be installed into a mobile phone working on Android platform. The user experiences no delays while using the program, which is a clear indication that the speed requirement is met. We made sure that the user interface simple and straight forward to use. In applications, where access control is vital, our application can be used to authenticate the sender of a message.

Also it is possible to detect, if the message has been corrupted or tampered with during transmission. Most importantly, the messages containing delicate information are stored securely and remain undisclosed even when the device is accessed by an adversary. The most unique and vital point to be considered is the security of the encrypted data against various attacks such as Brute Force attack, pattern attack etc. This application guarantees secure end to end transfer of data without any corrupt data segments.

6.REFERENCES

[1] J.Daemen and V.Rijmen, AES Proposal: Rijndael, NIST's AES home page, http ://www:nist:gov/aes."Announcing the Advanced Encryption Standard (AES)",Federal Information Processing Standards Publication 197, November 2001

[2] Priyanka Pimpale, Rohan Rayarikar and Sanket Upadhyay, "Modifications to AES Algorithm for Complex Encryption", IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.10, October 2011.

[3] Hassinen M.: SafeSMS 1.0 user manual. October 2004,Department of Computer Science, University of Kuopio.

[4] http://www.cs.uku.fi/ ~mhassine/SafeSMS/Manual en.pdf

[5] G. Racherla, D. Saha, "Security and Privacy Issues in Wireless and Mobile Computing", Proceedings of 2000 IEEE International Conference on Personal Wireless Communications, Dec 17-20, 2000, pp.509-513.

[6] [6]H. Marko, H. Konstantin, "Strong Mobile Authentication",Proceedings of 2nd International Symposium on WirelessCommunication Systems, Sept 5-7 2005, pp.96-100.

[7] Xinmiao Zhang and Keshab K. Parhi, "Implementation Approaches for the Advanced Encryption Standard Algorithm", 1531-636X/12, IEEE 2002.

[8] Chun Yan,Yanxia Guo, "A Research and Improvement Based on Rijndael Algorithm", 2009 First International Conference on Information Science and Engineering,Nanjing, Jiangsu China, December 26-December 28, ISBN:978-0-7695-3887-7

[9] Advanced Encryption Standard, http://en.wikipedia.org/wiki/Advanced_Encryption_Standard