ABSTRACT

The rise of internet has created a major challenge for securing sensitive information sent over the network. Several solutions have been proposed in the past to protect user data. Encryption algorithm is one of them. At present, various types of encryption algorithms like AES, DES, RSA and others are available. In this report, we have proposed an algorithm for ourselves which is 128-bit text size and 128-bit key size. We have compared the proposed algorithm with different algorithm comparisons. The result is showing our proposed algorithm gives better performance. By the difference we see that our recommended encryption process's average runtime is 0.006 sec which is less than other algorithms (AES, DES). It also protects various cryptanalytic attacks - key attack,brute force attack, man in the middle attack and it is also suitable for the implementation of software and hardware.

ABOUT THE AUTHOR

This project is done by **Suhas S** along with the help and support of many others. I thank all the people who have helped me to successfully execute this project within a given time period. This project is run on android studio Electric Eel or greater along with a the essential run environment.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Encryption is changing the way of information is displayed so that it is masked and the only way its true form can be viewed with a clear set of instruction. In its simple sense,Encryption is the process of encoding all user data or information using symmetric or asymmetrickey in such a way that even if an unauthorized party tries to access the data without keys, they won't be able to read it.

Decryption is the reverse process of encryption. It is the process of decoding all encrypted data using symmetric or asymmetric key in such a way that only authorized parties can access the data and can read it.

Data Encryption Standard (DES) is a symmetric key block cipher. In DES the key length is 112 bits or 168 bits and block size is 64-bits length. Now a day the increasing computational power is available which makes DES weak. For this reason, it can be attack by Brute Force Attacks other cryptanalytic attacks. In triple DES algorithm the size of block and key increased.

Without any doubt, AES is the strongest algorithm ever because it supports any combination of data and key length. In AES the number of round is variable and depends on length of the key. It uses 10 rounds for 128 bit keys, 12 rounds for 192 bit keys, 14 rounds for 256bit keys that can be divide into 4 basic operational blocks. These blocks are considered as array of bytes and organized as a matrix of the order of 4*4, which is called as state and subject to rounds where various transformations are done.

The XOR logical functions can be applied to binary bits and also considered as an encryption cipher. In the encryption context the strength of XOR cipher depends on the length andthe nature of the key. The XOR cipher with a lengthy random key can ensure better security performance. But large XOR key increases unpredictability and can confront brute force attack.

1.1 AIM AND OBJECTIVE

The main objective to design the android-based mobile is

1. To develop an algorithm for data security in an android platform.
2. To maintain the security of the system by preventing cryptanalytic attack like key attacks and brute force attacks.

CHAPTER 2

LITERATURE SURVEY

Cioc.I.Bet .al in 2015 explained a method used for increasing the security of sending text messages using public text communication services like email and SMS. It utilizes content encryption before sending the message through email or cell phone (SMS), so, even the message is received and seen by another unapproved individual, it can't be comprehended. So, that application was executed in LabVIEW and can be used for sending encoded content emailbetween at least two clients, utilizing open email administrations. For encryption, their proposed application utilizes content encryption strategy like balanced and deviated encryption, utilizing private encryption key or private or open encryption key. For sending encoded SMS using that application, the instant message must be recently scrambled, and after that the encoded message will be replicated to the content window of the application for sending SMS running on the cell phone. A comparable application can be additionally created for cell phones with working frameworks like Android, iOS, Windows portable and so forth their application can be utilized likewise with any instant message administration, similar to yippee detachment, Facebook Messenger and so on

Rayarikar Rohan, Upadhyay Sanket, Pimpale Priyanka 2012 have built up an application on android stage which enables the client to encode the messages before it is transmitted over the WAN network. They have utilized the propelled encryption standard (AES) calculation for encryption and unscrambling of the information. Their application can keep running on any gadget which takes a shot at android stage. This application gives a safe, quick and solid encryption of the information. There is a tremendous measure of perplexity and dissemination of the information during encryption which makes it exceptionally hard for an assailant to decipher the encryption design and the plain content structure the encryption information. The message encoded by the created application is likewise impervious to savage power and example assault. The different implementation of their application are in real life, and its usefulness are clarified in this paper. Majumder Jayeeta, Das Sagarjit, Maitysayak in 2015 explained on their report an application on android platform which allows the user to encrypt the messages before it istransmitted over the network. Their application can run on any device which works on android platform. Buba P.Z, Wajiga G.M in 2011 presented new cryptographic algorithms that employ theuse of asymmetric keys. The proposed algorithm decode message into nonlinear conditions

utilizing open key and unravel by the expected party utilizing private key. In the event that an outsider caught the message, it will be hard to decode it dew to the staggered figures of the proposed application. XOR does NOT produce entropy, it simply saves it. In particular one arbitrary Bitvector (various bits of fixed size) XORed with a nonrandom Bitvector rises to anotherirregular Bitvector. This is the premise behind applying keys to plaintext in PC based cryptography. Utilizing the basic OR would not work here as it would not be conceivable to reestablish the plaintext by ORing it once more, neither would AND work: Let (0,0,1,1,0) be the plaintext vector and (0,1,0,1,1) the key vector, then: OR: would result in (0, 1, 1, 1, 1) (this is our encryption step), ORing it again - to decrypt it- with the key would however result in (0, 1, 1, 1,1) which is not at all what our plaintext was originally. AND: would result in (0, 0, 0, 1, 0), let's try to decrypt it again: (0, 0, 0, 1, 0) so here too our plaintext would be lost forever. XOR:(finally) gives us: (0, 1, 1, 0, 1) and applying it again results in: (0, 0, 1, 1, 0) which is again our plaintext. Qualities returned by a hash function are called message review or just hash esteems. S-box is produced by deciding the multiplicative inverse for a given number in Rijndael's Galois Field.The multiplicative converse is then changed utilizing a relative change lattice. Where the segment is controlled by the least critical nybble, and the column is dictated by the most noteworthy nybble. The inverse S-box is simply the Sbox run in reverse

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 HARDWARE REQUIREMENTS

- 64-bit Microsoft® Windows® 8/10/11:
- x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor.
- 8 GB RAM or more.
- 8 GB of available disk space minimum (IDE + Android SDK + Android Emulator)
- 1280 x 800 minimum screen resolution

3.2 SOFTWARE REQUIREMENTS

- **Android studio version electric Eel**

In Android Studio Electric Eel, the new version of Logcat is enabled by default to make it easier to parse, query, and keep track of logs. This represents the most significant update to the tool since its introduction.

- **Google pixel Android Emulator**

A google pixel emulator simulates the functions of a Google Pixel device on the developer or tester's computer. This enables them to test an application on multiple google pixel devices without accessing each physical device.

- **SDK (Software Development Kit)**

SDK is the acronym for “Software Development Kit”. The SDK brings together a group of tools that enable the programming of mobile applications.This set of tools can be divided into 3 categories: SDKs for programming or operating system environments (iOS, Android, etc.), Application maintenance SDKs, Marketing and advertising SDKs

- **Gradle for Simulation**

Android Studio uses Gradle, an advanced build toolkit, to automate and manage the build process while letting you define flexible, custom build configurations.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

Systems analysis is the process by which an individual (s) studies a system such that an information system can be analyzed, modeled, and a logical alternative can be chosen.

4.1 PROBLEM STATEMENT AND SCOPE OF THE PROJECT

The importance of protecting sensitive information on an Android platform. A reliable security system is an absolute necessity in light of the growing likelihood that sensitive information will be compromised or that unauthorized users will gain access to it. The goal of this project is to design an algorithm that protects the secrecy of data and thwarts cryptanalytic assaults like key attacks and brute force attacks. The expanding sense of urgency regarding safety issues and the critical nature of preventing unauthorized access to sensitive information are the driving forces behind the need for this research. The investigation will cover a wide range of fields, such as business and non-commercial entities, banking transactions, and online shopping platforms. The purpose of the proposed cryptographic algorithm is to protect user data, allow for secure data sharing among authorized users, and do all of this while reducing the likelihood of data being stolen or lost.

4.2 SYSTEM ANALYSIS

System Analysis of APPLICATION USING DIFFERENT ENCRYPTION METHODS WITH CRYPTOGRAPHIC ALGORITHM using Android Studio:

4.2.1 Functional Requirements:

1. **User Authentication:** To guarantee that only authorized users may access the encrypted data, the system must offer a secure user authentication process.
2. **Data Encryption:** To protect user data confidentiality, the system must use reliable symmetric and asymmetric encryption techniques.
3. **Data Decryption:** The system must enable the decryption of encrypted data using theproper keys, enabling authorised parties to obtain and see the original data.
4. **Performance and Efficiency:** The system should be built to perform encryption and decryption operations quickly, with the least amount of processing overhead and best system performance.

4.2.2 Non-functional Requirements:

The following non-functional requirements need to be met for the game to be user- friendly and enjoyable:

1. **User Interface*:*** *The app has a responsive user interface that works well on different screen sizes and resolutions.*
2. **Stability:** *The app is made stable and reliable, with no crashes or glitches.*
3. **Responsiveness*:*** *The game is fast and responsive, with no noticeable lag or delay when using*
4. **Visual Appeal:** *The game is visually appealing, with an attractive design and appropriate graphics.*

4.2.3 Constraints:

The following constraints need to be taken into consideration when developing the Tic Tac Toe game using Android Studio:

Development Platform: The Application is to be developed using Android Studio.

Compatibility: The game must be compatible with Android devices running Android version 4.4 (KitKat) or higher.

Screen Size: The game must be able to run on devices with varying screen sizes and resolutions.

4.3 SYSTEM DESIGN

The system design for an Android APPLICATION USING DIFFERENT ENCRYPTION METHODS WITH CRYPTOGRAPHIC ALGORITHM can be divided into several components, each responsible for a specific aspect of the apps functionality. These components are:

4.3.1 User Interface (UI):

The UI component of the Application is responsible for creating an interactive and user-friendly interface for the player. It includes screens for the game start, game play, and game end, and it should be designed with an easy-to-use layout, appropriate graphics, and intuitive controls.

4.3.2 System Design Model

1. **User Authentication Model:**
    - To confirm users' identities before giving access to the system, implement a secure user authentication mechanism, such as username/password, biometric authentication, or multi-factor authentication.
2. **Encryption And Decryption Model:**

- Encrypt user data using powerful encryption methods, such as AES (Advanced Encryption Standard).
- For effective data encryption and decryption, use symmetric key encryption.
- Ensure that symmetric and asymmetric encryption methods are supported.
- Make sure that keys are generated, stored, and distributed properly.
- Incorporate decryption capabilities so that authorized users can retrieve and read encrypted data.

4.4 SYSTEM ARCHITECTURE

4.4.1 METHODOLOGY

These phases are planning, requirement analysis, proposed algorithm, design and development, testing and result.

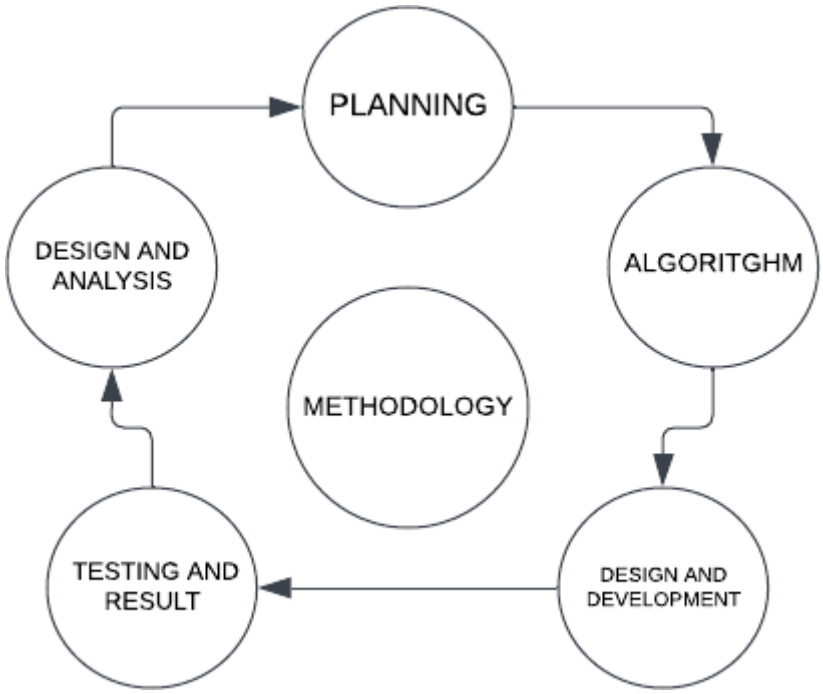

Fig.4.1: Methodology

4.4.2 PLANNING

For a successful work, pre-planning is essential because there are some sequential steps in every planning. We have done our plan by analyzing some journals and reports. There are many algorithms that have been created based on cryptography which could not fully complete cryptographic goal. Our target is to create a new cryptographic algorithm to achieve the cryptographic goal and that can be implemented in android platform.

4.4.3 ALGORITHM

Initially, algorithm starts with plain text. So, the plain text is act as input. Then the plain text transmitted into cipher text using several fixed encryption process randomly. In this method, the key generation process will be there. That means, the receiver must use the key to access the message that the sender will send. Therefore, to convert any plaintext into cipher text,we have to apply some fixed encryption process.

On the other hand, to convert any cipher text to plain text we have to apply some fixed decryption process. So, our main focus of this proposed algorithm is to protect all the information, that is being transmitted between sender and receiver without any data lose and to protect this information from unauthorized source

4.4.4 DESIGN AND DEVELOPMENT

After planning proposed model of encryption, we design the encryption and decryption process, which will satisfy with the authentication technique. We also design our required flowchart and for the proposed model. Then we start to developing the model with the required programming languages in android platform.

4.4.5 TESTING AND RESULT

After the development of our algorithm, we test the environment with some fixed data and compare our algorithm with other cryptographic algorithm like AES, DES, BlowFish with that fixed data.

4.4.6 DESIGN AND ANALYSIS

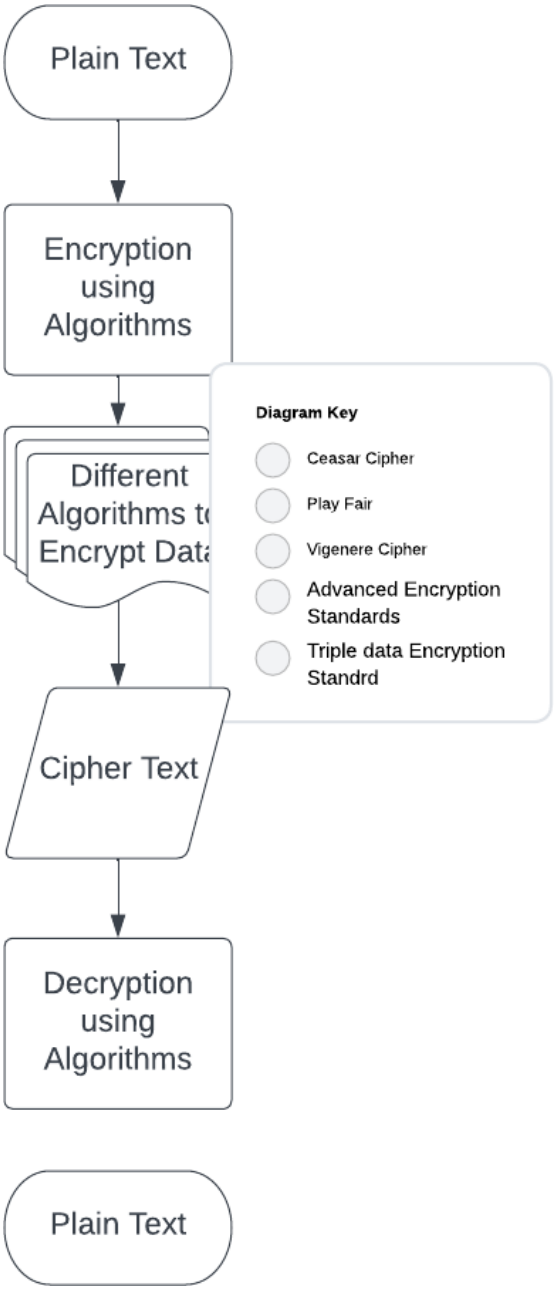

Fig.4.2 Architecture of Encryption and Decryption

CHAPTER 5

IMPLEMENTATION

IMPLEMENTING XML FILE

The first process to develop a APPLICATION USING DIFFERENT ENCRYPTION METHODS WITH CRYPTOGRAPHIC ALGORITHM is to design the layout file for the Android application. The layout is described using XML tags and attributes. The root element is a Relative Layout with some attributes specifying its size and margins. The background is set to an image (img) defined inthe resources of the app.

5.1.1 Main XML file

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="Main.Old_MainActivity">

  <FrameLayout
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

5.1.2 Splash Activity XML

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <ImageView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop"/>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
```

```
  android:orientation="vertical"
  android:scaleType="centerCrop"
  android:gravity="center">
  <ImageView
     android:id="@+id/SplashPC"
     android:layout_width="230dp"
     android:layout_height="230dp"
     android:layout_gravity="center" />



  </LinearLayout>
</FrameLayout>
```

5.1.3 Encryption Main XML

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <attr format="boolean" name="barrierAllowsGoneWidgets"/>
  <attr format="enum" name="barrierDirection">
     <enum name="left" value="0"/>
     <enum name="right" value="1"/>
     <enum name="top" value="2"/>
     <enum name="bottom" value="3"/>
     <enum name="start" value="5"/>
     <enum name="end" value="6"/>
  </attr>
  <attr format="boolean" name="chainUseRtl"/>
  <attr format="reference" name="constraintSet"/>
  <attr format="string" name="constraint_referenced_ids"/>
  <attr format="reference" name="content"/>
  <attr name="emptyVisibility">
     <enum name="gone" value="0"/>
     <enum name="invisible" value="1"/>
  </attr>
  <attr format="boolean" name="layout_constrainedHeight"/>
  <attr format="boolean" name="layout_constrainedWidth"/>
  <attr format="integer" name="layout_constraintBaseline_creator"/>
  <attr format="reference|enum" name="layout_constraintBaseline_toBaselineOf">
     <enum name="parent" value="0"/>
  </attr>
  <attr format="integer" name="layout_constraintBottom_creator"/>
  <attr format="reference|enum" name="layout_constraintBottom_toBottomOf">
     <enum name="parent" value="0"/>
  </attr>
  <attr format="reference|enum" name="layout_constraintBottom_toTopOf">
     <enum name="parent" value="0"/>
```

```
</attr>
<attr format="reference" name="layout_constraintCircle"/>
<attr format="integer" name="layout_constraintCircleAngle"/>
<attr format="dimension" name="layout_constraintCircleRadius"/>
<attr format="string" name="layout_constraintDimensionRatio"/>
<attr format="reference|enum" name="layout_constraintEnd_toEndOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="reference|enum" name="layout_constraintEnd_toStartOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="dimension" name="layout_constraintGuide_begin"/>
<attr format="dimension" name="layout_constraintGuide_end"/>
<attr format="float" name="layout_constraintGuide_percent"/>
<attr name="layout_constraintHeight_default">
   <enum name="spread" value="0"/>
   <enum name="wrap" value="1"/>
   <enum name="percent" value="2"/>
</attr>
<attr format="dimension|enum" name="layout_constraintHeight_max">
   <enum name="wrap" value="-2"/>
</attr>
<attr format="dimension|enum" name="layout_constraintHeight_min">
   <enum name="wrap" value="-2"/>
</attr>
<attr format="float" name="layout_constraintHeight_percent"/>
<attr format="float" name="layout_constraintHorizontal_bias"/>
<attr format="enum" name="layout_constraintHorizontal_chainStyle">
   <enum name="spread" value="0"/>
   <enum name="spread_inside" value="1"/>
   <enum name="packed" value="2"/>
</attr>
<attr format="float" name="layout_constraintHorizontal_weight"/>
<attr format="integer" name="layout_constraintLeft_creator"/>
<attr format="reference|enum" name="layout_constraintLeft_toLeftOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="reference|enum" name="layout_constraintLeft_toRightOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="integer" name="layout_constraintRight_creator"/>
<attr format="reference|enum" name="layout_constraintRight_toLeftOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="reference|enum" name="layout_constraintRight_toRightOf">
   <enum name="parent" value="0"/>
</attr>
<attr format="reference|enum" name="layout_constraintStart_toEndOf">
   <enum name="parent" value="0"/>
</attr>
```

```
  <attr format="reference|enum" name="layout_constraintStart_toStartOf">
     <enum name="parent" value="0"/>
  </attr>
  <attr format="integer" name="layout_constraintTop_creator"/>
  <attr format="reference|enum" name="layout_constraintTop_toBottomOf">
     <enum name="parent" value="0"/>
  </attr>
  <attr format="reference|enum" name="layout_constraintTop_toTopOf">
     <enum name="parent" value="0"/>
  </attr>
  <attr format="float" name="layout_constraintVertical_bias"/>
  <attr format="enum" name="layout_constraintVertical_chainStyle">
     <enum name="spread" value="0"/>
     <enum name="spread_inside" value="1"/>
     <enum name="packed" value="2"/>
  </attr>
  <attr format="float" name="layout_constraintVertical_weight"/>
  <attr name="layout_constraintWidth_default">
     <enum name="spread" value="0"/>
     <enum name="wrap" value="1"/>
     <enum name="percent" value="2"/>
  </attr>
  <attr format="dimension|enum" name="layout_constraintWidth_max">
     <enum name="wrap" value="-2"/>
  </attr>
  <attr format="dimension|enum" name="layout_constraintWidth_min">
     <enum name="wrap" value="-2"/>
  </attr>
  <attr format="float" name="layout_constraintWidth_percent"/>
  <attr format="dimension" name="layout_editor_absoluteX"/>
  <attr format="dimension" name="layout_editor_absoluteY"/>
  <attr format="dimension" name="layout_goneMarginBottom"/>
  <attr format="dimension" name="layout_goneMarginEnd"/>
  <attr format="dimension" name="layout_goneMarginLeft"/>
  <attr format="dimension" name="layout_goneMarginRight"/>
  <attr format="dimension" name="layout_goneMarginStart"/>
  <attr format="dimension" name="layout_goneMarginTop"/>
  <attr name="layout_optimizationLevel">
     <flag name="none" value="0"/>
     <flag name="standard" value="7"/> <!-- direct, barriers, chains -->
     <flag name="direct" value="1"/>
     <flag name="barrier" value="2"/>
     <flag name="chains" value="4"/>
     <flag name="dimensions" value="8"/>
     <flag name="groups" value="32"/>
  </attr>
  <declare-styleable name="ConstraintLayout_Layout"><attr name="android:orientation"/><attr
name="android:minWidth"/><attr name="android:minHeight"/><attr
name="android:maxWidth"/><attr name="android:maxHeight"/><attr
name="layout_optimizationLevel"/><attr name="constraintSet"/><attr
```

name="barrierDirection"/><attr name="barrierAllowsGoneWidgets"/><attr
name="constraint_referenced_ids"/><attr name="chainUseRtl"/><attr
name="layout_constraintCircle"/><attr name="layout_constraintCircleRadius"/><attr
name="layout_constraintCircleAngle"/><attr name="layout_constraintGuide_begin"/><attr
name="layout_constraintGuide_end"/><attr name="layout_constraintGuide_percent"/><attr
name="layout_constraintLeft_toLeftOf"/><attr name="layout_constraintLeft_toRightOf"/><attr
name="layout_constraintRight_toLeftOf"/><attr name="layout_constraintRight_toRightOf"/><attr
name="layout_constraintTop_toTopOf"/><attr name="layout_constraintTop_toBottomOf"/><attr
name="layout_constraintBottom_toTopOf"/><attr
name="layout_constraintBottom_toBottomOf"/><attr
name="layout_constraintBaseline_toBaselineOf"/><attr
name="layout_constraintStart_toEndOf"/><attr name="layout_constraintStart_toStartOf"/><attr
name="layout_constraintEnd_toStartOf"/><attr name="layout_constraintEnd_toEndOf"/><attr
name="layout_goneMarginLeft"/><attr name="layout_goneMarginTop"/><attr
name="layout_goneMarginRight"/><attr name="layout_goneMarginBottom"/><attr
name="layout_goneMarginStart"/><attr name="layout_goneMarginEnd"/><attr
name="layout_constrainedWidth"/><attr name="layout_constrainedHeight"/><attr
name="layout_constraintHorizontal_bias"/><attr name="layout_constraintVertical_bias"/><attr
name="layout_constraintWidth_default"/><attr name="layout_constraintHeight_default"/><attr
name="layout_constraintWidth_min"/><attr name="layout_constraintWidth_max"/><attr
name="layout_constraintWidth_percent"/><attr name="layout_constraintHeight_min"/><attr
name="layout_constraintHeight_max"/><attr name="layout_constraintHeight_percent"/><attr
name="layout_constraintLeft_creator"/><attr name="layout_constraintTop_creator"/><attr
name="layout_constraintRight_creator"/><attr name="layout_constraintBottom_creator"/><attr
name="layout_constraintBaseline_creator"/><attr name="layout_constraintDimensionRatio"/><attr
name="layout_constraintHorizontal_weight"/><attr name="layout_constraintVertical_weight"/><attr
name="layout_constraintHorizontal_chainStyle"/><attr
name="layout_constraintVertical_chainStyle"/><attr name="layout_editor_absoluteX"/><attr
name="layout_editor_absoluteY"/></declare-styleable>
<declare-styleable name="ConstraintLayout_placeholder"><attr name="emptyVisibility"/><attr
name="content"/></declare-styleable>
<declare-styleable name="ConstraintSet"><attr name="android:orientation"/><attr
name="android:id"/><attr name="android:visibility"/><attr name="android:alpha"/><attr
name="android:elevation"/><attr name="android:rotation"/><attr name="android:rotationX"/><attr
name="android:rotationY"/><attr name="android:scaleX"/><attr name="android:scaleY"/><attr
name="android:transformPivotX"/><attr name="android:transformPivotY"/><attr
name="android:translationX"/><attr name="android:translationY"/><attr
name="android:translationZ"/><attr name="android:layout_width"/><attr
name="android:layout_height"/><attr name="android:layout_marginStart"/><attr
name="android:layout_marginBottom"/><attr name="android:layout_marginTop"/><attr
name="android:layout_marginEnd"/><attr name="android:layout_marginLeft"/><attr
name="android:layout_marginRight"/><attr name="layout_constraintCircle"/><attr
name="layout_constraintCircleRadius"/><attr name="layout_constraintCircleAngle"/><attr
name="layout_constraintGuide_begin"/><attr name="layout_constraintGuide_end"/><attr
name="layout_constraintGuide_percent"/><attr name="layout_constraintLeft_toLeftOf"/><attr
name="layout_constraintLeft_toRightOf"/><attr name="layout_constraintRight_toLeftOf"/><attr
name="layout_constraintRight_toRightOf"/><attr name="layout_constraintTop_toTopOf"/><attr
name="layout_constraintTop_toBottomOf"/><attr name="layout_constraintBottom_toTopOf"/><attr
name="layout_constraintBottom_toBottomOf"/><attr
name="layout_constraintBaseline_toBaselineOf"/><attr

```
name="layout_constraintStart_toEndOf"/><attr name="layout_constraintStart_toStartOf"/><attr
name="layout_constraintEnd_toStartOf"/><attr name="layout_constraintEnd_toEndOf"/><attr
name="layout_goneMarginLeft"/><attr name="layout_goneMarginTop"/><attr
name="layout_goneMarginRight"/><attr name="layout_goneMarginBottom"/><attr
name="layout_goneMarginStart"/><attr name="layout_goneMarginEnd"/><attr
name="layout_constrainedWidth"/><attr name="layout_constrainedHeight"/><attr
name="layout_constraintHorizontal_bias"/><attr name="layout_constraintVertical_bias"/><attr
name="layout_constraintWidth_default"/><attr name="layout_constraintHeight_default"/><attr
name="layout_constraintWidth_min"/><attr name="layout_constraintWidth_max"/><attr
name="layout_constraintWidth_percent"/><attr name="layout_constraintHeight_min"/><attr
name="layout_constraintHeight_max"/><attr name="layout_constraintHeight_percent"/><attr
name="layout_constraintLeft_creator"/><attr name="layout_constraintTop_creator"/><attr
name="layout_constraintRight_creator"/><attr name="layout_constraintBottom_creator"/><attr
name="layout_constraintBaseline_creator"/><attr name="layout_constraintDimensionRatio"/><attr
name="layout_constraintHorizontal_weight"/><attr name="layout_constraintVertical_weight"/><attr
name="layout_constraintHorizontal_chainStyle"/><attr
name="layout_constraintVertical_chainStyle"/><attr name="layout_editor_absoluteX"/><attr
name="layout_editor_absoluteY"/><attr name="barrierDirection"/><attr
name="constraint_referenced_ids"/><attr name="android:maxHeight"/><attr
name="android:maxWidth"/><attr name="android:minHeight"/><attr
name="android:minWidth"/><attr name="barrierAllowsGoneWidgets"/><attr
name="chainUseRtl"/></declare-styleable>
  <declare-styleable
name="LinearConstraintLayout"><attr
name="android:orientation"/></declare-styleable>
</resources>
```

5.1.4 Fragment XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout

  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context="Main.MainActivity"
  android:id="@+id/drawer">

  <ImageView
     android:id="@+id/imageView"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:scaleType="centerCrop"
      />

  <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
```

android:orientation="vertical"

```
    android:padding="15dp"
    android:gravity="center">


    <LinearLayout
      android:id="@+id/containers"
      android:layout_marginTop="15dp"
       style="?android:attr/buttonBarButtonStyle"
       android:layout_width="250dp"
       android:layout_height="wrap_content"
       android:onClick="goToEncryption"
       android:background="@drawable/buttonshape"
       android:padding="10dp"

       android:layout_marginBottom="15dp">
       <TextView
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="Encryption"
          android:paddingStart="5dp"
          android:textStyle="bold"
          android:paddingEnd="5dp"

          android:textSize="40sp"
          android:textColor="@android:color/black"
          />
    </LinearLayout>


  </LinearLayout>



</androidx.drawerlayout.widget.DrawerLayout>
```

5.1.5 Algorithm Main XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android
  "xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/ConstraintLayout"
  android:background="#FFFFFF"
  tools:context="Main.">

  <ImageView
```

```
    android:id="@+id/imageView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:scaleType="centerCrop"

    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/Swtich"
    android:layout_width="300dp"
    android:layout_height="90dp"
    android:layout_marginTop="30dp"
    android:onClick="HashButtonClick"
    android:text="MD5"
    android:textColor="#000000"
    android:background="@drawable/buttonshape"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />


<EditText
    android:id="@+id/TextArea"
    android:layout_width="300dp"
    android:layout_height="160dp"
    android:layout_marginTop="20dp"
    android:background="@drawable/shape"
    android:gravity="center"
    android:hint="@string/enter_your_message_here"

    android:inputType="textMultiLine"
    android:textColor="#000000"
    android:textSize="15sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.504"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Swtich" />

<EditText
    android:id="@+id/salt"
    android:layout_width="208dp
    "
    android:layout_height="63dp"
    android:layout_marginTop="8dp"
    android:background="@drawable/shape"
    android:gravity="center"
    android:hint="Salt"
```

```
    android:maxLength="10"
    android:paddingTop="5dp"
    android:paddingBottom="5dp"
    android:textColor="#000000"
    android:textSize="15sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/TextArea" />

 <TextView
    android:id="@+id/Answer"
    android:layout_width="300dp"
    android:layout_height="160dp"
    android:layout_marginTop="12dp"
    android:background="@drawable/shape"
    android:gravity="center"
    android:hint="@string/your_output_gonna_be_here"
    android:inputType="textMultiLine"
    android:textColor="#000000"
    android:textSize="15sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.495"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/hash_Buuton"
    tools:ignore="TextViewEdits" />

 <Button
    android:id="@+id/hash_Buuton"
    android:layout_width="111dp"
    android:layout_height="60dp"
    android:layout_marginTop="20dp"
    android:background="@drawable/buttonshape"
    android:onClick="HashButtonClick"
    android:text="@string/hash"
    android:textSize="19sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/salt" />

 <Button
    android:id="@+id/copy_button"
    android:layout_width="111dp"
    android:layout_height="60dp"
    android:background="@drawable/buttonshape"
    android:onClick="encryptionButtonClick"
    android:text="@string/copy"
    android:textSize="20sp"
    app:layout_constraintBottom_toTopOf="@+id/Matrix"
    app:layout_constraintEnd_toStartOf="@+id/reset_button"
```

```
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/Answer" />

  <Button
    android:id="@+id/reset_button"
    android:layout_width="111dp"
    android:layout_height="60dp"
    android:background="@drawable/buttonshape"
    android:onClick="encryptionButtonClick"
    android:text="@string/reset"
    android:textSize="20sp"
    app:layout_constraintBottom_toTopOf="@+id/Matrix"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/copy_button"
    app:layout_constraintTop_toBottomOf="@+id/Answer" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

5.2 JAVA FILE

5.2.1 main activity java

```
package Main;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import com.example.Algorithms.R;
import Encryption.EncryptionMain;
public class MainActivity extends AppCompatActivity {
  EncryptionMain encryptionMain;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Fragment fragment = new MainFragment();
    FragmentManager fragmentManager = getSupportFragmentManager();
    fragmentManager.beginTransaction().replace(R.id.container, fragment).commit(); }
  public void goToEncryption(View view) {
    encryptionMain = new EncryptionMain();
    FragmentManager manager = getSupportFragmentManager();
    FragmentTransaction transaction = manager.beginTransaction();
    transaction.setCustomAnimations(android.R.anim.fade_in, android.R.anim.fade_out,
android.R.anim.fade_in, android.R.anim.fade_out);
    transaction.replace(R.id.container, encryptionMain);
    transaction.addToBackStack(null);
    transaction.commit(); }
  public void encryptionButtonClick(View view) {
```

```
    try {
        switch (view.getId()) {
            case R.id.Swtich:
                encryptionMain.switchAlgho(view);
                break;
            case R.id.Encrypt_Buuton:
                encryptionMain.encrypt(view);
                break;
            case R.id.Decrypt_Buuton:
                encryptionMain.decrypt(view);
                break;
            case R.id.copy_button:
                encryptionMain.copyToClipboard(view);
                break;
            case R.id.reset_button:
                encryptionMain.reset(view)
                ;break;         }
    } catch (Exception e) {
        e.printStackTrace();  }  }}
```

5.2.2 Main Fragement java

```
 package Main;
import android.os.Bundle;
import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import com.example.Algorithms.R;
import Encryption.EncryptionMain;
public class MainActivity extends AppCompatActivity {
  EncryptionMain encryptionMain;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Fragment fragment = new MainFragment();
    FragmentManager fragmentManager = getSupportFragmentManager();
    fragmentManager.beginTransaction().replace(R.id.container, fragment).commit();  }
  public void goToEncryption(View view) {
    encryptionMain = new EncryptionMain();
    FragmentManager manager = getSupportFragmentManager();
    FragmentTransaction transaction = manager.beginTransaction();
    transaction.setCustomAnimations(android.R.anim.fade_in, android.R.anim.fade_out,
android.R.anim.fade_in, android.R.anim.fade_out);
    transaction.replace(R.id.container, encryptionMain);
    transaction.addToBackStack(null);
    transaction.commit(); }
  public void encryptionButtonClick(View view) {
    try {
```

```
        switch (view.getId()) {
            case R.id.Swtich:
                encryptionMain.switchAlgho(view);
                break;
            case R.id.Encrypt_Buuton:
                encryptionMain.encrypt(view);
                break;
            case R.id.Decrypt_Buuton:
                encryptionMain.decrypt(view);
                break;
            case R.id.copy_button:
                encryptionMain.copyToClipboard(view);
                break;
            case R.id.reset_button:
                encryptionMain.reset(view)
                ;break;         }
      } catch (Exception e) {
          e.printStackTrace();
      }
   }
}
```

5.2.3 Splash activity java

```
package com.ajstudios.easyattendance;
package Main;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;
import com.example.Algorithms.R;
public class SplashActivity extends AppCompatActivity {
   private int Splash_Screen = 3000;
   private ImageView LogoImg;
   private TextView Appname;
   private Animation topAnimation, leftAnimation;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_splash);
      getSupportActionBar().hide();
      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
      LogoImg = findViewById(R.id.SplashPC);
      //Appname = findViewById(R.id.SplashKeyboard);
```

```
    topAnimation = AnimationUtils.loadAnimation(this, R.anim.top_animation);
    //leftAnimation = AnimationUtils.loadAnimation(this, R.anim.left_aimation);
    LogoImg.setAnimation(topAnimation);
    //Appname.setAnimation(leftAnimation);
    new Handler().postDelayed(new Runnable() {
      @Override
      public void run() {
        startActivity(new Intent(SplashActivity.this, MainActivity.class));
        finish();
      }
    },Splash_Screen);
  }
}
```

5.2.4 Encryption Main java

```
package Encryption;
import android.content.Context;
import android.os.Bundle;
import android.text.InputType;
import android.util.Base64;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.constraintlayout.widget.ConstraintSet;
import androidx.fragment.app.Fragment;
import com.example.Algorithms.R;
import java.nio.charset.StandardCharsets;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PublicKey;
import Encryption.Algorithms.AES;
import Encryption.Algorithms.DES;
import Encryption.Algorithms.PlayFair;
import Encryption.Algorithms.Vigenere;
import Encryption.Algorithms.Caesarcipher;
public class EncryptionMain extends Fragment {
  private String message;
  private String key;
  private Button Switch;
  private Button Encrypt_Buuton;
```

```
  private Button Decrypt_Buuton;
  private PlayFair p;
  private TextView Answer;
  private EditText Textfield_Text;
  private EditText Textfield_Key;
  private TextView Matrix_value;
  private TextView Play_Fair_VALUE;
  private View view;
  @Override
  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
      view = inflater.inflate(R.layout.encryption_main, container, false);
      ((AppCompatActivity) getActivity()).getSupportActionBar().hide();
      getActivity().getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
      Switch = view.findViewById(R.id.Swtich);
      Encrypt_Buuton = view.findViewById(R.id.Encrypt_Buuton);
      Decrypt_Buuton = view.findViewById(R.id.Decrypt_Buuton);
      Answer = view.findViewById(R.id.Answer);
      Textfield_Text = view.findViewById(R.id.TextArea);
      Textfield_Key = view.findViewById(R.id.Key);
      Matrix_value = view.findViewById(R.id.Matrix);
      Play_Fair_VALUE = view.findViewById(R.id.Play_Fair_VALUE);
      return view; }
  public void encrypt(View view) throws Exception {
      if (Textfield_Text.length() == 0) {
          Toast.makeText(view.getContext(), "Enter a message to Encrypt",
Toast.LENGTH_SHORT).show();
          return;
      }
      message = String.valueOf(Textfield_Text.getText());
      key = String.valueOf(Textfield_Key.getText());
      String Algorithm = String.valueOf(Switch.getText());
      switch (Algorithm) {
          case "Advanced Encryption Standard":
              AES aes = new AES();
              String enc = aes.AESencrypt(key.getBytes("UTF-16LE"), message.getBytes("UTF-16LE"));
              Answer.setText(enc);
              break;
          case "Triple Data Encryption Standard":
              DES des = new DES();
              String encData = des.encrypt(key.getBytes("UTF-16LE"), message.getBytes("UTF-16LE"));
              Answer.setText(encData);
              break;
          case "Caesar Cipher":
              if (key.isEmpty()) {
                  Toast.makeText(view.getContext(), "Enter a key to Encrypt",
Toast.LENGTH_SHORT).show();
                  return;
              }
```

```
            if (key.length() > 26) {
                Toast.makeText(view.getContext(), "The Key must be less than 26 characters",
Toast.LENGTH_SHORT).show();
                return;
            }
            Caesarcipher c = new Caesarcipher();
            Answer.setText(c.caesarcipherEnc(message, Integer.parseInt(key)));
            break;
        case "Vigenere Cipher":
            if (Textfield_Key.length() == 0) {
                Toast.makeText(view.getContext(), "Enter a key to Encrypt",
Toast.LENGTH_SHORT).show();
                return;
            }
            for (char i : message.toUpperCase().toCharArray()) {
                if (i < 'A' || i > 'Z') {
                    Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
                    return;
                }
            }
            for (char i : key.toUpperCase().toCharArray()) {
                if (i < 'A' || i > 'Z') {
                    Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
                    return;
                }
            }
            Vigenere v = new Vigenere();
            Answer.setText(v.Vigenereencrypt(message, key));
            break;
        case "Play Fair":
            try {
                p = new PlayFair("");
                Play_Fair_VALUE.setText(p.Encrypt(message, key));
                Matrix_value.setText(p.getT1());
            } catch (Exception e) {
                Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
            }
            break;
    }
  }
  public void decrypt(View view) throws Exception {
    if (Textfield_Text.length() == 0) {
        Toast.makeText(view.getContext(), "Enter a message to Decrypt",
Toast.LENGTH_SHORT).show();
        return;
    }
    message = String.valueOf(Textfield_Text.getText());
```

```
    key = String.valueOf(Textfield_Key.getText());
    String SwitchValue = Switch.getText().toString();
    switch (SwitchValue) {
        case "Advanced Encryption Standard":
            AES aes = new AES();
            try {
                String decData = aes.AESdecrypt(key, Base64.decode(message.getBytes("UTF-16LE"),
Base64.DEFAULT));
                Answer.setText(decData);
            } catch (Exception e) {
                Toast.makeText(view.getContext(), "Your key is wrong",
Toast.LENGTH_SHORT).show();
            }
            break;
        case "Triple Data Encryption Standard":
            DES des = new DES();
            try {
                String decData = des.decrypt(key, Base64.decode(message.getBytes("UTF-16LE"),
Base64.DEFAULT));
                Answer.setText(decData);
            } catch (Exception e) {
                Toast.makeText(view.getContext(), "Your key is wrong",
Toast.LENGTH_SHORT).show();
            }
            break;
        case "Caesar Cipher":
            if (Textfield_Key.length() == 0) {
                Toast.makeText(view.getContext(), "Enter a key", Toast.LENGTH_SHORT).show();
                return;             }
            if (Integer.parseInt(key) >= 26) {
                Toast.makeText(view.getContext(), "The Key must be less than 26 characters",
Toast.LENGTH_SHORT).show();
                return;             }
            Caesarcipher c = new Caesarcipher();
            Answer.setText(c.caesarcipherDec(message, Integer.parseInt(key)));
            break;
        case "Vigenere Cipher":
            if (Textfield_Key.length() == 0) {
                Toast.makeText(view.getContext(), "Enter a key to Decrypt",
Toast.LENGTH_SHORT).show();
                return;             }

            for (char i : message.toUpperCase().toCharArray()) {
                if (i < 'A' || i > 'Z') {
                    Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
                    return;                 }             }
            for (char i : key.toUpperCase().toCharArray()) {
                if (i < 'A' || i > 'Z') {
```

```
                Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
                return;                }
                                       }Vigenere
            v = new Vigenere();
            Answer.setText(v.Vigeneredecrypt(message, key));
            break;
        case "Play Fair":
            try {
                Play_Fair_VALUE.setText(p.Decrypt(message, key));
                Matrix_value.setText(p.getT1());
            } catch (Exception e) {
                Toast.makeText(view.getContext(), "Only Letters are allowed here",
Toast.LENGTH_SHORT).show();
            }
            break;      }  }
  public void switchAlgho(View view) {
      reset(null);
      String SwitchValue = Switch.getText().toString();
      switch (SwitchValue) {
          case "Advanced Encryption Standard":
              Switch.setText("Triple Data Encryption Standard");
              break;
          case "Triple Data Encryption Standard":
              Textfield_Key.setInputType(InputType.TYPE_CLASS_NUMBER);
              Switch.setText("Caesar Cipher");
              break;
          case "Caesar Cipher":
              Textfield_Key.setInputType(InputType.TYPE_CLASS_TEXT);
              Switch.setText("Vigenere Cipher");
              break;
          case "Vigenere Cipher":
              Textfield_Key.setVisibility(View.VISIBLE);
              Answer.setVisibility(View.GONE);
              Matrix_value.setVisibility(View.VISIBLE);
              Play_Fair_VALUE.setVisibility(View.VISIBLE);
              Switch.setText("Play Fair");
              break;
          case "Play Fair":
              Answer.setVisibility(View.VISIBLE);
              Matrix_value.setVisibility(View.GONE);
              Play_Fair_VALUE.setVisibility(View.GONE);
              Switch.setText("Advanced Encryption Standard");
              break;      } }
  public void reset(View view) {
      Textfield_Text.setText("");
      Textfield_Key.setText("");
      Answer.setText("");
      Play_Fair_VALUE.setText("");
      Matrix_value.setText("");
      if(view!=null)
```

```
    Toast.makeText(view.getContext(), "All data has been deleted", Toast.LENGTH_SHORT).show();
}
  public void copyToClipboard(View view) {
    if (Play_Fair_VALUE.length() == 0) {
       String copyText = String.valueOf(Answer.getText());
       if (Answer.length() == 0) {
          Toast.makeText(view.getContext(), "There is no message to copy",
Toast.LENGTH_SHORT).show()
          ;return;        }
       int sdk = android.os.Build.VERSION.SDK_INT;
       if (sdk < android.os.Build.VERSION_CODES.HONEYCOMB) {
          android.text.ClipboardManager clipboard = (android.text.ClipboardManager)
               view.getContext().getSystemService(Context.CLIPBOARD_SERVICE);
          clipboard.setText(copyText);
       } else {
          android.content.ClipboardManager clipboard = (android.content.ClipboardManager)
               view.getContext().getSystemService(Context.CLIPBOARD_SERVICE);
          android.content.ClipData clip = android.content.ClipData
               .newPlainText("Your message :", copyText);
          clipboard.setPrimaryClip(clip);          }
       Toast.makeText(view.getContext(),
            "Your message has be copied", Toast.LENGTH_SHORT).show();
    } else {
       int sdk = android.os.Build.VERSION.SDK_INT;
       if (sdk < android.os.Build.VERSION_CODES.HONEYCOMB) {
          android.text.ClipboardManager clipboard = (android.text.ClipboardManager)
               view.getContext().getSystemService(Context.CLIPBOARD_SERVICE);
          clipboard.setText(Play_Fair_VALUE.getText().toString());
       } else {
          android.content.ClipboardManager clipboard = (android.content.ClipboardManager)
               view.getContext().getSystemService(Context.CLIPBOARD_SERVICE);
          android.content.ClipData clip = android.content.ClipData
               .newPlainText("Your message :", Play_Fair_VALUE.getText().toString());
          clipboard.setPrimaryClip(clip);          }          Toast.makeText(view.getContext(), "Your
message has be copied", Toast.LENGTH_SHORT).show();      }  }}
```

5.2.5 Algorithms java

AES algorithm

```
package Encryption.Algorithms;
import
android.util.Base64;
import android.util.Log;
import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class AES {
public String AESencrypt ( byte[] key, byte[] clear) throws Exception {
       MessageDigest md = MessageDigest.getInstance("md5");
```

```
        byte[] digestOfPassword = md.digest(key);
        SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return Base64.encodeToString(encrypted, Base64.DEFAULT);
    }public String AESdecrypt (String key,byte[] encrypted) throws Exception {
    MessageDigest md = MessageDigest.getInstance("md5");
    byte[] digestOfPassword = md.digest(key.getBytes("UTF-16LE"));
    SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES"); Cipher cipher =
    Cipher.getInstance("AES/ECB/PKCS7Padding"); cipher.init(Cipher.DECRYPT_MODE,
    skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return new String(decrypted, "UTF-16LE");
    }

}
```

Caesar Cipher algorithm

```
package Encryption.Algorithms;
import
android.util.Base64;
import android.util.Log;
import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class AES {
        public String AESencrypt ( byte[] key, byte[] clear) throws Exception {
        MessageDigest md = MessageDigest.getInstance("md5"); byte[] digestOfPassword =
md.digest(key);
        SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES");
        cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);

        return Base64.encodeToString(encrypted, Base64.DEFAULT);
    }
    pubic String AESdecrypt (String key,byte[] encrypted) throws Exception
    {MessageDigest md = MessageDigest.getInstance("md5");
    byte[] digestOfPassword = md.digest(key.getBytes("UTF-16LE"));
        SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS7Padding");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return new String(decrypted, "UTF-16LE");
    }
```

```
}
```

DES algorithm

```
package Encryption.Algorithms;
import
android.util.Base64;
import android.util.Log;
import java.security.MessageDigest;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
public class DES {
        public String encrypt ( byte[] key, byte[] clear) throws Exception {
       MessageDigest md = MessageDigest.getInstance("md5");
       byte[] digestOfPassword = md.digest(key);
       SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "DESede");
       Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
       cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
       byte[] encrypted = cipher.doFinal(clear);
       return Base64.encodeToString(encrypted, Base64.DEFAULT);
   }
   public String decrypt (String key,byte[] encrypted) throws Exception
   {MessageDigest md = MessageDigest.getInstance("md5");
       byte[] digestOfPassword = md.digest(key.getBytes("UTF-16LE"));

       SecretKeySpec skeySpec = new SecretKeySpec(digestOfPassword, "DESede");
       Cipher cipher = Cipher.getInstance("DESede/CBC/PKCS5Padding");
       cipher.init(Cipher.DECRYPT_MODE, skeySpec);
       byte[] decrypted = cipher.doFinal(encrypted);
       return new String(decrypted, "UTF-16LE");
   }
}
```

Playfair cipher

```
package Encryption.Algorithms;
public class PlayFair {
private String t1="";
  public String getT1() {
     return t1;
  }
  public PlayFair(String t1) {
     this.t1 = t1;
  }
```

```
public class Basic {
   String allChar = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   boolean indexOfChar(char c) {
      for (int i = 0; i < allChar.length(); i++) {
         if (allChar.charAt(i) == c)
            return true;
      }
      return false;      } }
Basic b = new Basic();
char keyMatrix[][] = new char[5][5];
boolean repeat(char c) {
   if (!b.indexOfChar(c))
      {return true;    }
   for (int i = 0; i < keyMatrix.length; i++) {
      for (int j = 0; j < keyMatrix[i].length; j++) {
         if (keyMatrix[i][j] == c || c == 'J')
            return true;         }
                                  }
   return false; }
public void insertKey(String key) {
   key = key.toUpperCase();
   key = key.replaceAll("J", "I");
   key = key.replaceAll(" ", "");
   int a = 0, b = 0;
   for (int k = 0; k < key.length(); k++) {
      if (!repeat(key.charAt(k))) {
         keyMatrix[a][b++] = key.charAt(k);
         if (b > 4) {
            b = 0;
            a++;
         }
      }
   }

   char p = 'A';

   while (a < 5) {
      while (b < 5) {
         if (!repeat(p)) {
            keyMatrix[a][b++] = p;

         }
         p++;
      }
      b = 0;
      a++;
   }

   for (int i = 0; i < 5; i++) {

      for (int j = 0; j < 5; j++) {
```

```
        }
    }
    t1=t1.concat("\n");
    for(int i=0;i < 5;i++)
    {
        for(int j=0;j < 5;j++)
            t1 = t1 +"  "+ keyMatrix[i][j];

             t1=t1.concat("\n");          }
                                          }int
rowPos(char c) {
    for (int i = 0; i < keyMatrix.length; i++) {
        for (int j = 0; j < keyMatrix[i].length; j++) {
            if (keyMatrix[i][j] == c)
                return i;
        }
    }
    return -1;
}

int columnPos(char c) {
    for (int i = 0; i < keyMatrix.length; i++) {
        for (int j = 0; j < keyMatrix[i].length; j++) {
            if (keyMatrix[i][j] == c)
                return j;
        }
    }
    return -1;
}

public String encryptChar(String plain) {
    plain = plain.toUpperCase();
    char a = plain.charAt(0), b = plain.charAt(1);
    String cipherChar = "";
    int r1, c1, r2, c2;
    r1 = rowPos(a);
    c1 = columnPos(a);
    r2 = rowPos(b);
    c2 = columnPos(b);

    if (c1 == c2) {
        ++r1;
        ++r2;
        if (r1 > 4)
            r1 = 0;

        if (r2 > 4)
            r2 = 0;
        cipherChar += keyMatrix[r1][c2];
        cipherChar += keyMatrix[r2][c1];
```

```
    } else if (r1 == r2) {
        ++c1;
        ++c2;
        if (c1 > 4)
            c1 = 0;

        if (c2 > 4)
            c2 = 0;
        cipherChar += keyMatrix[r1][c1];
        cipherChar += keyMatrix[r2][c2];

    } else {
        cipherChar += keyMatrix[r1][c2];
        cipherChar += keyMatrix[r2][c1];
    }
    return cipherChar;
}


public String Encrypt(String plainText, String key) {
    insertKey(key);
    String cipherText = "";
    plainText = plainText.replaceAll("j", "i");
    plainText = plainText.replaceAll(" ", "");
    plainText = plainText.toUpperCase();
    int len = plainText.length();

    if (len / 2 != 0) {
        plainText += "X";
        ++len;
    }

    for (int i = 0; i < len - 1; i = i + 2) {
        cipherText += encryptChar(plainText.substring(i, i + 2));
        cipherText += " ";
    }
    return cipherText;
}


public String decryptChar(String cipher) {
    cipher = cipher.toUpperCase();
    char a = cipher.charAt(0), b = cipher.charAt(1);
    String plainChar = "";
    int r1, c1, r2, c2;
    r1 = rowPos(a);
    c1 = columnPos(a);
    r2 = rowPos(b);
    c2 = columnPos(b);
```

```
    if (c1 == c2) {
        --r1;
        --r2;
        if (r1 < 0)
            r1 = 4;

        if (r2 < 0)
            r2 = 4;
        plainChar += keyMatrix[r1][c2];
        plainChar += keyMatrix[r2][c1];
    } else if (r1 == r2) {
        --c1;
        --c2;
        if (c1 < 0)
            c1 = 4;

        if (c2 < 0)
            c2 = 4;
        plainChar += keyMatrix[r1][c1];
        plainChar += keyMatrix[r2][c2];

    } else {
        plainChar += keyMatrix[r1][c2];
        plainChar += keyMatrix[r2][c1];
    }
    return plainChar;
  }


  public String Decrypt(String cipherText, String key) {
    String plainText = "";
    cipherText = cipherText.replaceAll("j", "i");
    cipherText = cipherText.replaceAll(" ", "");
    cipherText = cipherText.toUpperCase();
    int len = cipherText.length();
    for (int i = 0; i < len - 1; i = i + 2) {
        plainText += decryptChar(cipherText.substring(i, i + 2));
        plainText += " ";

    }
    return plainText;
  }


}
```

Vigener cipher

```
    package Encryption.Algorithms;
```

```
public class Vigenere {
  public String Vigenereencrypt (String text, String key)
  {

     String res = "";
     text = text.toUpperCase();
     key = key.toUpperCase();
     for (int i = 0, j = 0; i < text.length(); i++) {
        char c = text.charAt(i);
        if (c < 'A' || c > 'Z') continue;
        res += (char) ((c + key.charAt(j) - 2 * 'A') % 26 + 'A');
        j = ++j % key.length();
     }
     return res;

  }
  public String Vigeneredecrypt (String text, String key)
  {
     String res = "";
     text = text.toUpperCase();
     key = key.toUpperCase();
     for (int i = 0, j = 0; i < text.length(); i++) {
        char c = text.charAt(i);
        if (c < 'A' || c > 'Z') continue;
        res += (char) ((c - key.charAt(j) + 26) % 26 + 'A');
        j = ++j % key.length();
     }

     return res;

  }
}
```

CHAPTER 6

RESULTS

Fig 6.1 Main Page

Fig 6.2 Advanced encryption Standard Fragment

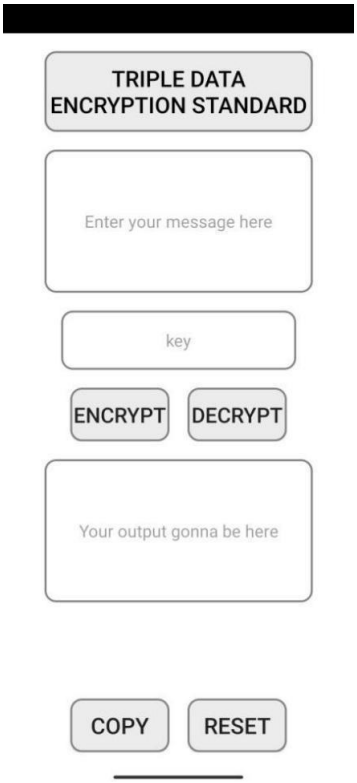

Fig 6.3 Triple data Encryption Standard

Fig 6.4 Caesar Cipher

Fig 6.5 PlayFair Cipher

Fig 6.6 Vigenere Cipher

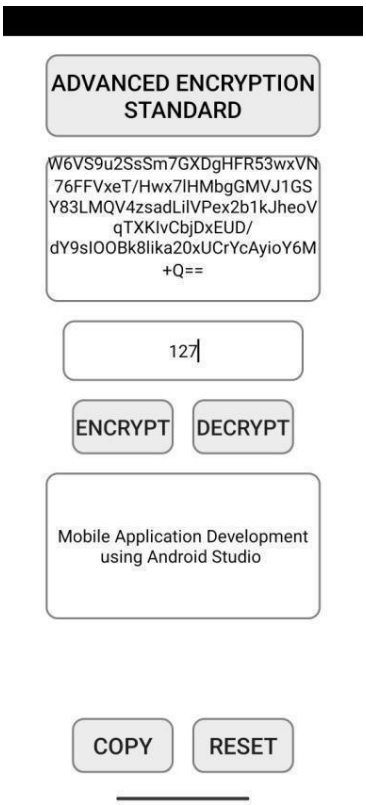

Fig 6.7 Advance Encryption Standard With Ciphertext

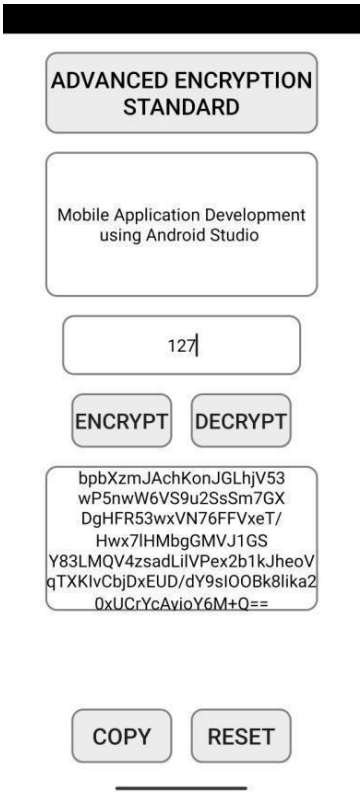

Fig 6.8 Advance Decryption Standard With Plaintext

Fig 6.9 PlayFair Cipher With Encrypted Data

The Above Figure from 6.1 represents the Main page of the Application. The Figures 6.2 to 6.6 represents the The Algorithms Present in the Application and it defines the different types of the Encryption And Decryption methods that can be used to Secure the data.

Starting the app. There are 2 options, these are – encryption and decryption. For encrypting a text, first we have to click on the Encryption button and enter the message which we want to encrypt. Enter a secret key and click on the Encryption button. After that we get an encrypted text which we call cipher text and also get a hash code. For decryption we have to copy our cipher text and hash code and back to the main menu and click on the Decryption button.

CHAPTER 7

CONCLUSION AND FUTURE WORK

The main purpose of our proposed algorithm is to secure any message or information without any loss of data. For this reason, it requires to provide some feature for senders and receivers. These features are known as cryptographic goal. These goals are – confidentiality, integrity, authentication and non-repudiation of data. We have tried to achieve all the goals of the cryptography in our algorithm. In this algorithm we have used symmetric key encryption which is private key based encryption. We have created an algorithm which basically works on block wise where every block can store 128-bit data. After various analyses we found that, it is very strong in terms of security and the speed.

- In future, the authors will work on multithreading as if each blocks can process at a time.
- Audio, image, video and file will be encrypted

CHAPTER 8

REFERENCES

[1] B. Cioc, M. Jurian, I. Lita, and R. M. Teodorescu, A method for increasing security in electronic communication services based on text messages communication, Electronics, Computers and Artificial Intelligence (ECAI), 2015 7th International Conference on, 2015, AE–23p.

[2] Z. P. Buba and G. M. Wajiga, Cryptographic algorithms for secure data communication, Int. J. Comput. Sci. Secure. IJCSS.2011.5(2):227– 243p.

[3] N. Mathur and R. Bansode, AES Based Text Encryption Using 12 Rounds with Dynamic Key Selection, Procedia Comput. Sci. 2017.79:1036– 1043p.

[4] M. Nagendra and M. C. Sekhar, Performance Improvement of Advanced Encryption Algorithm using Parallel Computation, Int. J. Softw. Eng. Its Appl., 2014:10p