# Android Video Encryption & Sharing

## A PROJECT REPORT

***Submitted by***

# Roushan Kumar (19bcs2896)

# Rakesh Chouhan (19bcs2987)

# Shivendra Kumar Singh (19bcs3107)

***in partial fulfillment for the award of degree of***

## BACHELOR OF ENGINEERING

**IN**

## COMPUTER SCIENCE & ENGINEERING

## 

**Chandigarh University**

Dec 2022



**BONAFIDE CERTIFICATE**

Certified that this project report “**Android Video Encryption & Sharing**” is the bonafide work of Roushan Kumar (19bcs2896) , Rakesh Chouhan (19bcs2987) , Shivendra Kumar Singh (19bcs3107) who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Dr. NAVPREET KAUR WALIA Er. ANUP LAL YADAV

## 

## 

## HEAD OF THE DEPARTMENT SUPERVISOR

## Submitted for the project viva-voce examination held on 17 Nov 2022

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**Er. ANUP LAL YADAV Er. SIMRAN**

**TABLE OF CONTENTS**

List of figures……………………………………………………………………v

Abstract………………………………………………………………………………………..vi

Graphical Abstract ……………………………………………………………………………viii

Abbreviations …….………………………………………………………………………….. ix

Symbols………………………………………………………………………………………..x

Chapter 1 : Introduction……………………………………………………………………….1

1.1 Client Identification/Need Identification/Identification of relevant Contemporary issue...1

1.2 Identification of Problem………………………………………………………………….1

1.3 Identification Of Task……………………………………………………………………..2

1.4 Timeline…………………………………………………………………………………...3

1.5 Orgination of Report………………………………………………………………………4

Chapter 2…………………………………………………………………………………………6

LITERATURE REVIEW/BACKGROUND STUDY…………………………………………...6

2.1. Timeline of the reported problem……………………………………………………………6

2.2 Proposed solutions……………………………………………………………………………7

2.3. Bibliometric analysis………………………………………………………………………...7

2.4. Review Summary……………………………………………………………………………8

2.5. Problem Definition…………………………………………………………………………..10

2.6. Goals/Objectives…………………………………………………………………………….11

Chapter 3………………………………… ……………………………………………………..14

Design and flow process…………………………………………………………………………14

3.1. Evaluation & Selection of Specifications/Features………………………………………….14

3.2. Design Constraints…………………………………………………………………………..21

3.3. Analysis and Feature finalization subject to constraints……………………………………..22

3.4. Design Flow…………………………………………………………………………………24

3.5. Design selection……………………………………………………………………………..25

3.6. Implementation plan/methodology………………………………………………………….27

CHAPTER 4…………………………………………………………………………………….30

Results Analysis And Validation………………………………………………………………..30

4.1. Implementation of solution…………………………………………………………………30

Use modern tools in: …………………………………………………………………………….31

• analysis, …………………………………………………………………………………..33

• design drawings/schematics/ solid models,……………………………………………….37

• report preparation, ………………………………………………………………………...40

• project management, and communication, ………………………………………………..45

• Testing/characterization/interpretation/data validation……………………………………49

Introduction to Activities…………………………………………………………………………52

CHAPTER 5……………………………………………………………………………………..53

Conclusion and Future Word…………………………………………………………………….53

5.1 Conclusion………………………………………………………………………………….55

5.2 Future work…………………………………………………………………………….…...55

References ……………………………………………………………………………………….56

Appendix…………………………………………………………………………………………61

User Manual……………………………………………………………………………………...67

**List Of Figures**

Fig 1. Graphical Abstract…………………………………………………..viii

Fig 2. Video Sharing Protocol………………………………………………16

Fig 3. User Interaction Protocol…………………………………………….19

Fig 4. Key Generation Technique…………………………………………..21

Fig 5. Flow Diagram………………………………………………………...21











**ABSTRACT**

Increased availability of mobile cameras has led to more opportunities for people to record videos of significantly more of their lives. Many times people want to share these videos, but only to certain people who were co-present. Since the videos may be of a large event where the attendees are not necessarily known, we need a method for proving co-presence without revealing information before co-presence is proven. In this demonstration, we present a privacy-preserving method for comparing the similarity of two videos without revealing the contents of either video. This technique leverages the Similarity of Simultaneous Observation technique for detecting hidden webcams and modifies the existing algorithms so that they are computationally feasible to run under fully homomorphic encryption scheme on modern mobile devices.

The demonstration will consist of a variety of devices preloaded with our software. We will demonstrate the video sharing software performing comparisons in real time. We will also make the software available to Android devices via a QR code so that participants can record and exchange their own videos

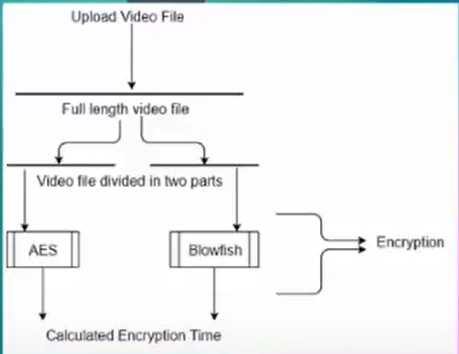
The Sharing and Surfing of Real Time Video project is quite similar to YouTube application in some functionality. The source video is uploaded by the user itself which undergoes through various process which takes care of video security. In this system, video will be saved in encrypted form and stored in database.

**सार**

रियल टाइम वीडियो प्रोजेक्ट की शेयरिंग और सर्फिंग कुछ कार्यक्षमता में YouTube एप्लिकेशन के समान है। स्रोत वीडियो उपयोगकर्ता द्वारा ही अपलोड किया जाता है जो वीडियो सुरक्षा का ख्याल रखने वाली विभिन्न प्रक्रियाओं से गुजरता है। इस प्रणाली में, वीडियो एन्क्रिप्टेड रूप में सहेजा जाएगा और डेटाबेस में संग्रहीत किया जाएगा। एईएस और ब्लोफिश एल्गोरिदम के रूप में एन्क्रिप्शन और उसी रूप में डिक्रिप्ट भी।

इस प्रणाली में, एक वीडियो को दो भागों में विभाजित किया जाता है, एक भाग एईएस रूप में एन्क्रिप्ट किया जाता है और दूसरा भाग ब्लोफिश एन्क्रिप्टेड प्रारूप में विभाजित होता है। दूसरे रूप में पूरा वीडियो एईएस एन्क्रिप्टेड फॉर्म में विभाजित होता है और फिर ब्लोफिश फॉर्म और डिक्रिप्टेड इसके विपरीत डाउनलोडिंग प्रक्रिया में होता है। एप्लिकेशन वीडियो के साझाकरण को बहुत सुरक्षित बनाता है जो इस विकसित एप्लिकेशन को दूसरों से विशिष्ट बनाता है।

**GRAPHICAL ABSTRACT**



**ABBREVIATIONS**

1. “H” for High
2. “V” for Variable
3. “L” for Low
4. “ENK” for Enryption
5. “DECR” for Decryption

**SYMBOLS**

1. “?” for Low
2. “√” for satisfied
3. “X” for not satisfied

**CHAPTER 1**

**INTRODUCTION**

In the early-1990s, when the commercial Internet was still young (!), security was taken seriously by most users. Many thought that increased security provided comfort to paranoid people while most computer professionals realized that security provided some very basic protections that we all needed? Cryptography for the masses barely existed at that time and was certainly not a topic of common discourse. By the turn of the century, of course, the Internet had grown in size and importance so as to be the provider of essential communication between billions of people around the world .

Encryption is the process of encoding all user data on an Android device using symmetric encryption keys. Once a device is encrypted, all user-created data is automatically encrypted before committing it to disk and all reads automatically decrypt data before returning it to the calling process.

* 1. **Client Identification/Need Identification/Identification of relevant Contemporary issue**

The wide use of digital images and videos in various applications brings serious attention to security and privacy issues today. Data encryption is a suitable method to protect data. Till now, various encryption algorithms have been proposed and widely used (DES, RSA, IDEA, AES etc.), most of which are used for text and binary data. It is difficult to use them directly in video encryption as video data are often of large volumes and require real time operations.

**1.2. Identification of Problem**

These are videos that a person would like to share with others who they may or may not know, but not do so indiscriminately. The goal of our system is to enhance the ability for users to share videos with people that they may or may not know, but only if those people were co-present when the video was taken. For example, Alice and Bob are attending a party for a mutual acquaintance, Carol, but did not meet each other at the party.

Alice and Bob do not know each other, but both of them take videos of the festivities. At a later time, Alice is looking for additional videos of the party because she is trying to create a video collage for Carol. Bob does not want to provide the video to Alice if she was not actually at the party and likewise, Alice does not want to reveal her video to Bob if he was not there. Our system enables Alice to use her video as proof to Bob that she was present without revealing any of the content of the video.



**Figure 1.1 Encrypting problem**

**1.3. Identification of Tasks**

No existing work meets the requirements of our video sharing system. The focus of some previous work has been on obscuring the video itself whereas we want all videos that are shared to be in original condition . Initially, the Ciphertext is passed through Blowfish algorithm in bytes format using the same key. Also, these systems are designed to release information to the user, albeit in an obfuscated form, even if they were not present at the event. In this system, video will be saved in encrypted form and stored in database. Encryption in form of AES and Blowfish algorithm and decrypted also in the same form. Other systems are designed to limit the sharing of video to known groups or users whereas we need videos to be distributed without having to have a pre-existing relationship with the other users.

In many cases, a cloud server performs the processing on plaintext data and breaks our requirement that an honest, but curious, server should also not learn the content of the video . We are able to accomplish this through the use of fully homomorphic encryption. There are so many android phones activating each day and through that phone’s so many videos are shared to each other. This enables our system to operate on encrypted data so that no information about the videos is revealed to either party.

**1.4. Timeline**

We will bring several devices such as laptops and mobile phones that have our software preloaded on them that we will use to demonstrate the video sharing software. In addition, we will also provide a QR code that will link to an APK that Android users will be able to install on their mobile phones so that they can also participate in the demo with their own devices and try it in other areas besides in front of our poster.

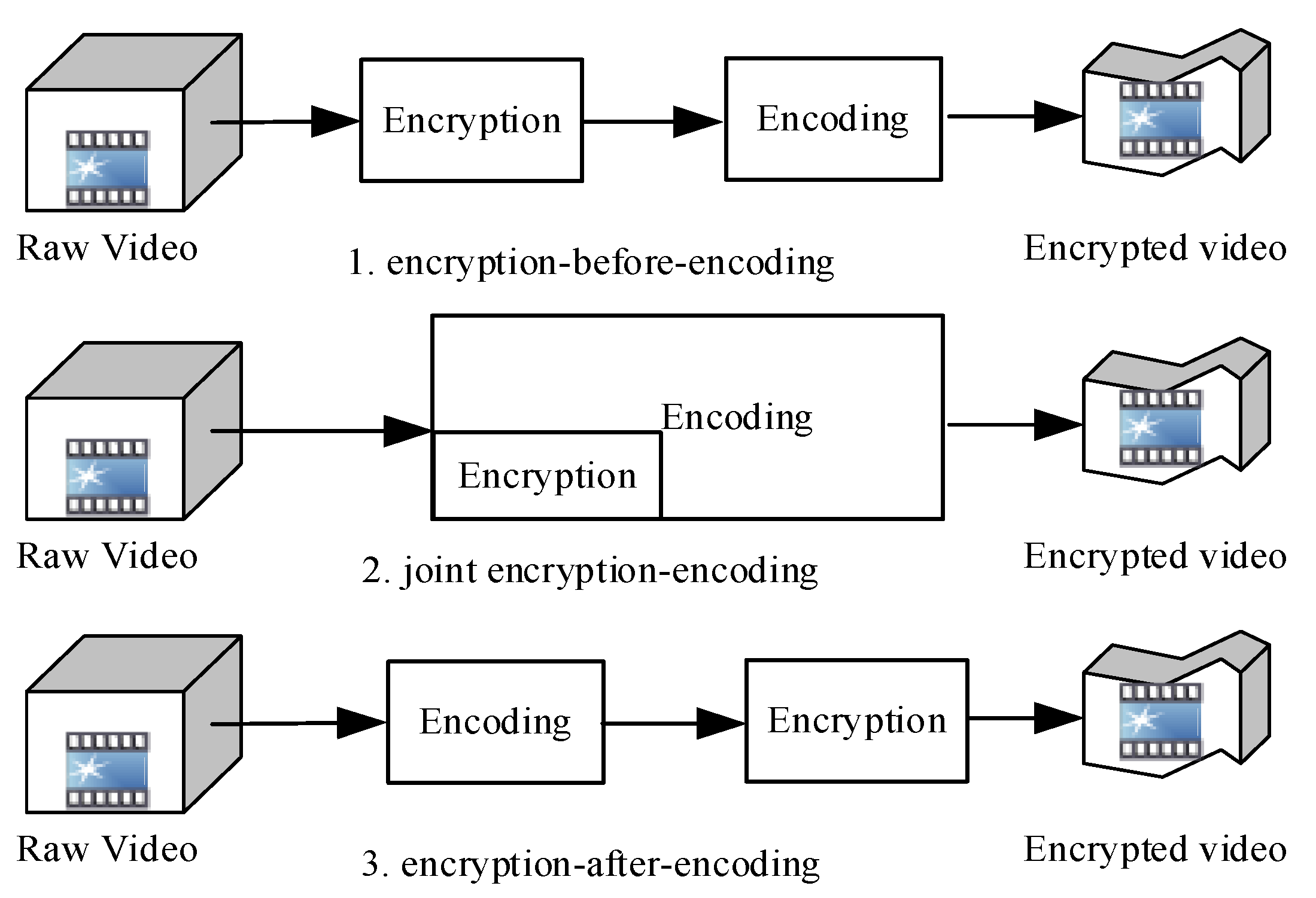
|  |  |  |  |
| --- | --- | --- | --- |
| ***Ref- Number*** |  | ***Working & Use of This Reviews*** | ***Objectives of This Work*** |
| 1 | Android Based on AES Algorithm | * The encryption key is 128-bit used in the AES algorithm. * Failure of statistical analysis and text encryption pattern due to the proposed algorithm components with strong deployment and confusion   The security feature of this algorithm and Altai considered more important is that there are no differential or linear attacks that can break this algorithm | * Secure and improved SMS application. * Encrypted information from message recipients is preserved. * Decrypt message as requested by users. * Provide protection against misuse of message information by the user.   Improved security with high confidentiality of data. |
| 2 | A novel approach | * The method has three points: encryption process, SMS gateway and decryption process. * In the encryption process, the message is encrypted based on the public key that is derived from the shared public key, in SMS gateways all of the messages is know because The definitions of mobile devices stored in the server and all keys are created by definitions, in the decryption steps, the device can easily   decrypt messages using its own key. | Scheme design makes users communicate among themselves efficiently without sharing or sharing unique keys. |
| 3 | Signature using IMEI of device, key secret and payload is GZIP and encoding Base64 | To enable Unicode string transfer over 7-bit SMS the output is encrypted with Base64. | * The evaluation of the security mechanism applied in the GSM live network between two Android devices showed that the proposed secure   communications have no effect on the time of SMS delivery compared to normal M2M calls via SMS.  It can provide better compression ratio. |
| 4 | sensors on the device is using to encrypted data stored and transmitted in mobile. | Using touch gestures implemented encrypted SMS messages | The encryption method used allows more secure encryption of data and then transfer over multiple channels. |
| 5 | Android Based on RSA Algorithm | Using touch gestures implemented encrypted SMS messages | * Execution this method is faster in the mobile and added another layer of security. * This paper tested in different   types of mobile types (GalaxyS3, Galaxy S4, HTC). |
| 6 | Android platform in AES algorithm in | * Analyzed the AES algorithm. There are total internal transformation (Sub Bytes, Shift Rows, Mixed Column and ADD Round Key).   Android platform, a package javax. crypto. chiper is called to make it easier in performing the coding. | * The different of testing are performed on applications are: * Compatibility testing and Interface testing. * Se Low level resource testing. * services testing and Performance testing. * Operational testing, Installation tests and Security Testing.   GUI (Graphical User Interface) |
| 7 | Android platform in NTRU Algorithm | * Key size of NTRU is Very small. * Key Generation of NTRU is 200 times. * Encryption of operation /sec of NTRU is 1113 times. * Decryption/sec of NTRU is 1132 */*ms for NTRU-251. * NTRU account strength is lower when compared to both mobile and smart cards. * Speed of NTRU is Faster.   Efficiency of NTRU is Fastest. | * Less execution times. * Decrease decoding time.   Output factor productivity to be less useful in the consumption of the mixture |
| 8 | ECIES or RSA-OAEP | * This application can be used for secure transportation.   Use authenticated and authentic encryption and short digital signatures that make gathering all the information in a text message essential for encrypted SMS authentication, while maintaining a useful length of messages. | * Flexible trade-offs between security and message length provide three levels of security:   1. confidentiality only, (2) confidentiality and message authentication, and (3) confidentiality, origin authentication and non- repudiation.   Providing easy-to-use security levels and seamless encryption integration are addressing the usability of security features. |
| 9 | Android platform in RC4 and AES with Rijndael encryption engine.  This paper simulation operation in paper [6]. | * Using RC4 and AES in Secure SMS Management Center includes two components:  1. The components and structure of the SEESMS program include the following modules:    * • Registration service (RS)    * Server Message Handler (SMH)    * Secure Storage (SS)    * Cryptosystem Engines (CE): 2. To exchange digitally signed and encrypted SMS. It includes the following parts.    * Message Handler (MH)    * Secure Storage (SS)    * Cryptosystem Engines   Keys Communicator | * CE are modules that secure messages that are exchanged with a remote user and each CE carries an encryption system and provides three standard functions: generation of key, message plaintext / cipher and message signing / verification. * Keys Communicator is the unit responsible for implementing the client's key exchange protocol, used to deliver client-generated encryption keys to SSMC. * Analysis of the results clearly shows that Rijndael has better encryption strength while RC4   provides better energy conservation. |
| 10 | Android platform in RC4 and AES with Rijndael encryption engine.  This paper simulation operation in paper . | * Using RC4 and AES in Secure SMS Management Center includes two components:  1. The components and structure of the SEESMS program include the following modules:    * • Registration service (RS)    * Server Message Handler (SMH)    * Secure Storage (SS)    * Cryptosystem Engines (CE): 2. To exchange digitally signed and encrypted SMS. It includes the following parts.    * Message Handler (MH)    * Secure Storage (SS)    * Cryptosystem Engines   Keys Communicator | * CE are modules that secure messages that are exchanged with a remote user and each CE carries an encryption system and provides three standard functions: generation of key, message plaintext / cipher and message signing / verification. * Keys Communicator is the unit responsible for implementing the client's key exchange protocol, used to deliver client-generated encryption keys to SSMC. * Analysis of the results clearly shows that Rijndael has better encryption strength while RC4   provides better energy conservation. |

**1.5. Organization of the Report**

Although an important and rich variety of video encryption algorithms have been proposed in literature, most of the algorithms defined in Table 1 are not secure against cryptanalysis attack. Naïve algorithm provides highest level of security but it is very slow in nature and cannot be used in real time. Permutation based algorithms are generally faster but they do not provide sufficient level of security.

Selective encryption algorithms reduces computational complexity by selecting only a minimal set of data to encrypt but their security and speed level generally vary based on which and how many parameters they encrypt.

Perceptual encryption algorithms are suitable for application like pay per view TV, video on demand where potential users like to see low quality video before buying them



**Figure 1.2 : Raw Video Encrypt Into Encrypted Video**

Login Activity consist of five total fields one image view two text fields one login button and a text view Register Activity. The two text fields are for Email and Password and a login button is there to take verified user to the next interface apart from that there is one Register Yourself Text view with take first time user to register themselves by email id and password.

|  |  |
| --- | --- |
| 1 | You will use Android studio to create an Android application under a package.myapplication. |
| 3 | Modify src/MainActivity.java file to add necessary code. |
| 4 | Modify the res/layout/activity main to add respective XML components |
| 5 | Run the application and choose a running android device and install the application on it and verify the results |

**CHAPTER 2**

**LITERATURE REVIEW/BACKGROUND STUDY**

**2.1. Timeline of the reported problem**

In , the authors proposed video encryption algorithm (VEA) which uses a secret key to randomly change the signs of all DCT coefficients in an MPEG stream. It is fast as it operates on a small portion of original video. It is more efficient than DES algorithm because it only selectively encrypts a small number of bits of the MPEG compressed video and selected bit is only XORed one time with the corresponding bit of the secret key. VEA does not protect from plaintext attack provided the attacker knowns the original video image (plaintext and ciphertext).

On an Android phone, the preprocessing time, which includes precomputation and encryption of all the data takes between 4 and 8 seconds on a Pixel 2 mobile phone to compare 60 seconds of video. While this is a large amount of time for pre-processing on the mobile phone, this is a process does not have to be run every time.

We cache the results after a video has been processed, so this is only a one time cost that can be incurred during the time between recording a video and sharing the video, so it is unlikely to be noticed by the user since a user is not expected to immediately share the video with a person that they are unsure was at the event. The similarity computations on average complete in 200-400 milliseconds on the mobile phone.

“A Modified AES for Mobile Devices” in 2015. This thesis modified the AES to encrypt data mobile phone. He takes different cases to show result of system between classical AES and modified AES in terms of computational complexity and security. The platform used in mobile and programming language is the Android Studio. The author claims the adjust AES encryption algorithms have many advantages which are; robust encryption, fast encryption and decryption process, and easy implementation.[6] • “Video Frame Encryption Algorithm using AES” in 2016, their methodology focuses on security and privacy of digital video. They have used mpeg video compression, encryption and decryption technique. These proposed in IJERT about this system needs some improvement, as AES alone is not much secure now-a-days

“Encryption using two algorithms” in 2014. In this paper the authors described the technique to encrypt the data and messages in mobile devices transmitted over network .This technique is developed under android platform and used two algorithms for encryption data , the first used symmetric AES and the second used asymmetric ECC , in sender and receiver sides are used the appropriate keys for encryption and decryption of the data .and he claims the system are achieving confidentiality ,authenticity ,and integrity of massage and data . “Video Encryption Using AES Algorithm” in 2014. This paper used Advanced Encryption Standard (AES) Algorithm for Video encryption. AES algorithm is also compared with a modified algorithm of the Data Encryption Standard (DES). The results referred that encryption and decryption time in AES is better.

**2.2. Proposed solutions**

Timing: On an Android phone, the preprocessing time, which includes precomputation and encryption of all the data takes between 4 and 8 seconds on a Pixel 2 mobile phone to compare 60 seconds of video. While this is a large amount of time for pre-processing on the mobile phone, this is a process does not have to be run every time. We cache the results after a video has been processed, so this is only a one time cost that can be incurred during the time between recording a video and sharing the video, so it is unlikely to be noticed by the user since a user is not expected to immediately share the video with a person that they are unsure was at the event .

AES-256 is the recommended key for top-notch security. Those with the proper encryption key can use it to reverse the encryption process and see the original unencrypted data. In general, AES-128 video encryption should be plenty secure for most use cases.

application for the Android based mobile devices. Although an important and rich variety of video encryption algorithms have been used, most of the algorithms defined are not completely secured. This application allows users to send file or data in the video format to other android devices in a secure network. In this way, application can still access the data without reaching for user’s sensitive information. This application can be useful in many real time events and applications for an enhanced user support. It is observed that the decryption process takes more time relatively as compared with the encryption process, which is acceptable because of the nature of the algorithms besides this the algorithms are implemented in limited resources. The main advantage of this system is achieving the protection for data of video in android devices such as confidentiality for the secure end-to-end user communication and for simple user interface so that it is easy for users to interact.

**2.3. Bibliometric analysis**

**A. Uploading the video file**: After login / registering user needs to select the video file from their feed and upload it or can share which they want to share. This video will be converted into bytes and stored in document format .

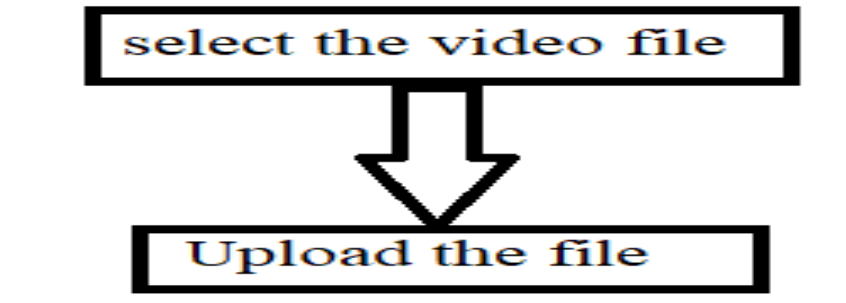


Figure 2. Upload the file in Videozen

**B. Division of file:** Using split function in java we have divide the file into two parts, for the ease of encryption process. While in decryption, this whole process is followed in reversed and the whole file is combined together and convert to video format.

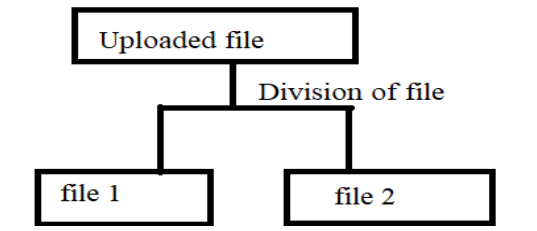


Figure: 2 Division of video files.

**C.Encrypting Process:** The encryption process undergoes two different algorithms that is AES and Blowfish. The first half of the video undergoes into AES and other part of the video undergoes into Blowfish. As we know, blowfish accepts inputs in bytes format for this purpose we converted input in bytes format. We have used AES with block size of 128 bits and key size for AES is 192 bits and for blowfish block size is 64bits and key size is 192 bits. This keys are stored in database. Firebase database is used for storing the data.

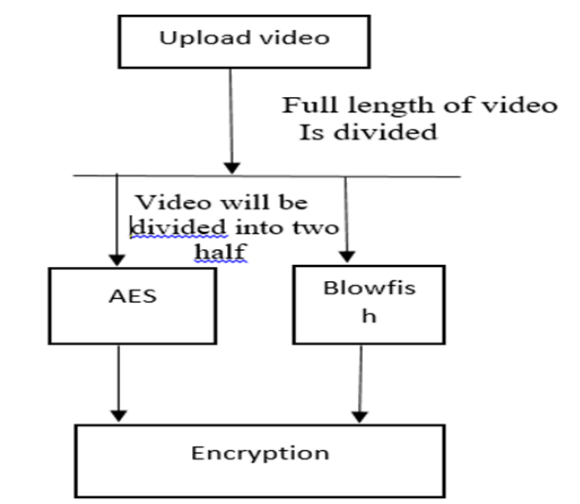


Figure: 3 Encrypting the video files

**D. Storing this encrypted file into database:** The files or the data of the video format after the encryption process are stored into the database.

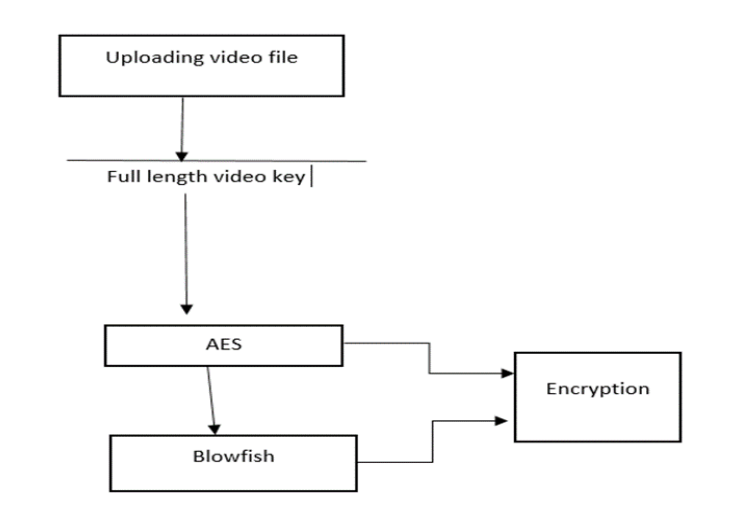
****

Fig: 4 Storing the file in database

**E. Decryption Process:** After the encryption process, the divided parts of the video are combined together to make to bring back the video into the original length. The reverse process of the encryption is carried out for decryption. In Decryption, the whole process is followed in reverse order. Initially, the Ciphertext is passed through Blowfish algorithm in bytes format using the same key. Then, that file is passed through AES algorithm using the key of AES which was used initially for encrypting the file. Then this file is divided in two parts among which one is passed through AES and Blowfish algorithm. After this whole decryption process the Users can download the whole combined video.

**2.4. Review Summary**

We need to define a set of parameters based on which we can evaluate and compare video encryption algorithms. Some parameters listed below are gathered from literature.

**Visual Degradation (VD)**: This criterion measures the perceptual distortion of the video data with respect to the plain video. In some applications, it could be desirable to achieve enough visual degradation, so that an attacker would still understand the content but prefer to pay to access the unencrypted content. However, for sensitive data, high visual degradation could be desirable to completely disguise the visual content.

**Encryption ratio (ER)**: This criterion measures the ratio between the size of encrypted part and the whole data size. Encryption ratio has to be minimized to reduce computational complexity.

In practical applications, for a video encryption algorithm, security, time efficiency, format compliance and compression friendliness are really important . Among them, security is the basic requirement, which means that the cost of breaking the encryption algorithm is no smaller than the ones buying the video’s authorization.

**Speed (S)**: In many real-time video applications, it is important that the encryption and decryption algorithms are fast enough to meet real time requirements.

**Compression Friendliness (CF)**: An encryption algorithm is considered compression friendly if it has no or very little impact on data compression efficiency. Some encryption algorithms impact data compressibility or introduce additional data that is necessary for decryption. It is desirable that size of encrypted data shoud not increase.

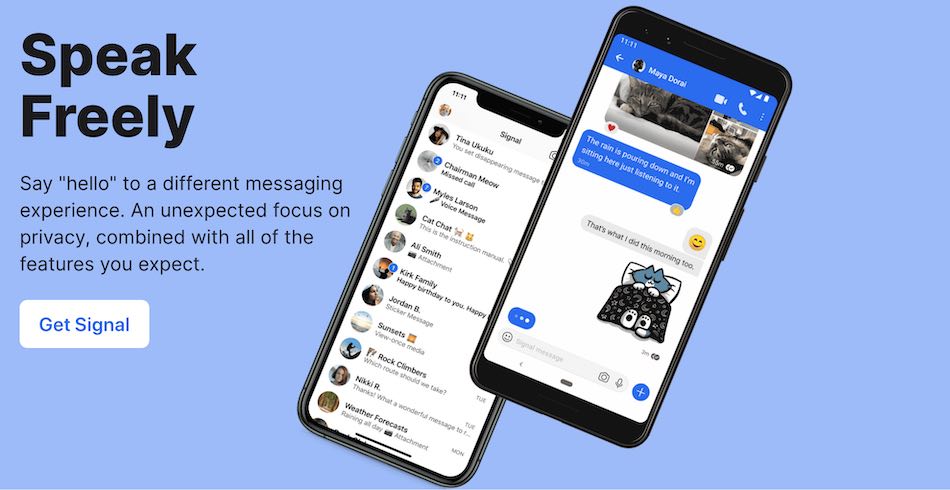
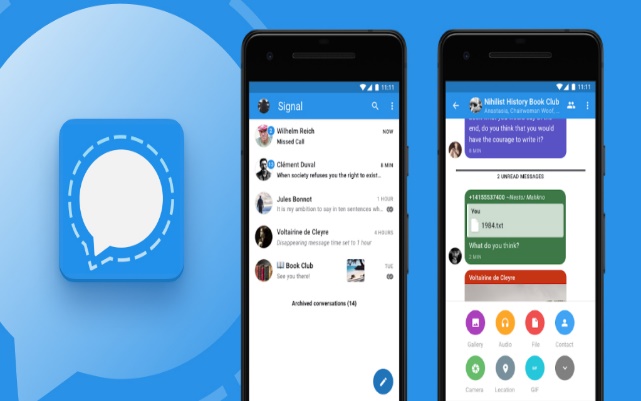
**Format Compliance (FC)**: The encrypted bit stream should be compliant with the compressor. And standard decoder should be able to decode the encrypted bit stream without decryption.

**Cryptographic Security (CS)**: Cryptographic security defines whether encryption algorithm is secure against brute force and different plaintext-ciphertext attack? For highly valuable multimedia application, it is really important that the encryption algorithm should satisfied cryptographic security. classification of video encryption algorithms is presented. Video encryption algorithms are classified based on their unique way of encrypting data.

classification of video encryption algorithms is presented. Video encryption algorithms are classified based on their unique way of encrypting data. presents performance parameters of these algorithms based on which evaluation and comparison is done. classification of video encryption algorithms is presented. Video encryption algorithms are classified based on their unique way of encrypting data. presents performance parameters of these algorithms based on which evaluation and comparison is done. classification of video encryption algorithms is presented. Video encryption algorithms are classified based on their unique way of encrypting data. Section 2.2 presents performance parameters of these algorithms based on which evaluation and comparison is done.

**2.4. Review Summary**

Videos are transferred through various types of computer network. To secure video communication different encryption methodologies are used. Due to huge size of digital videos they are generally transmitted in compressed formats such as MPEG 1/2/4[4][5][7], H.263/ H.264/AVC [6][7]. Various encryption algorithms have been proposed in literature. This section presents detail description of these algorithms.

**2.5. Problem Definition**

The most straightforward method to encrypt every byte in the whole Moving Picture Expert Group (MPEG) stream using standard encryption schemes such as DES or AES. The idea of naïve algorithm is to treat the MPEG bitstream as text data and does not use any of the special structure [2]. It provides the security to whole MPEG stream because every byte is encrypted, and no algorithm exists to break triple DES or AES so far.

It is not applicable solution for big video, because it is very slow especially when we use triple DES. Because of the encryption operation delay increases and overload will be unacceptable for real time video application. As encryption is performed after compression, no impact is observed on compression efficiency.

It is critical to write well-structured code when writing any piece of code, especially for larger programs. All code should be written in a way that makes it simple to follow and understand. To accomplish this, the programmer must be divided into various components, functions, and classes. After a high-level design has been created, each component can be written separately.

**2.6. Goals/Objectives**

In Zig-Zag permutation [11],instead of mapping the 8X8 block to 1X64 vector in “Zig-Zag” order, it maps individual 8x8 block to 1x64 vector by using a random permutation list (secret key). There are many ways to produce a permutation list which has uniform distribution over all possible permutations. This algorithm consists of three steps.

i)Generate a permutation list with cardinality 64.

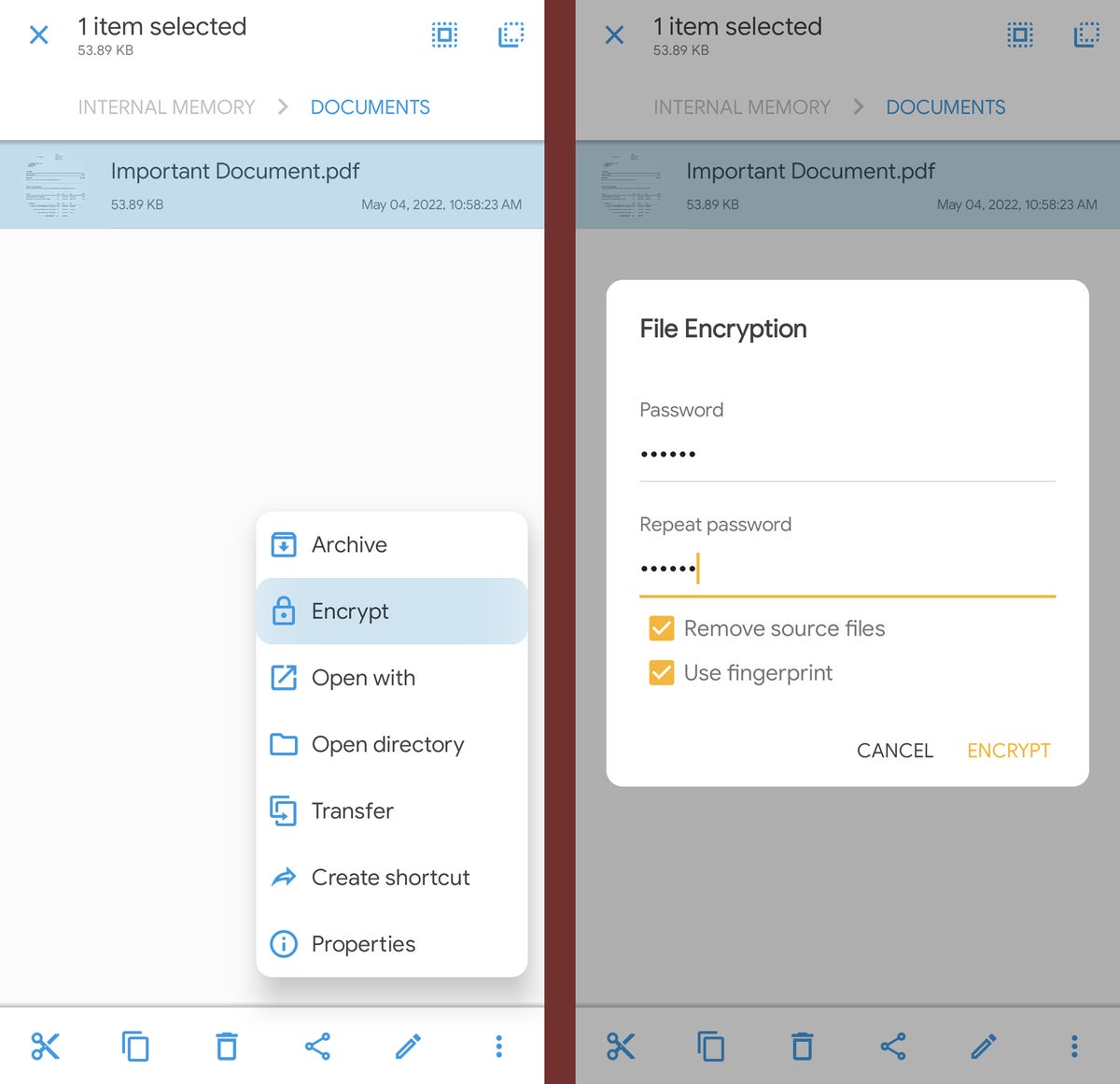
ii) Complete splitting procedure after 8x8 block is quantized.

iii) Apply the random permutation list to the split block, and pass the result to the entropy coding procedure. Since mapping Zig-Zag order and mapping according to random permutation list have the same computational complexity, the encryption and decryption add very little overhead to the video compression and decompression processes.

However, this method decreases the video compression rate because the random permutation distorts the probability distribution of Discrete Cosine Transform (DCT) coefficients and make Huffman table used less than optimal. Zig- Zag permutation algorithm cannot withstand the known-plaintext attack.

By assuming that we know certain frames of video in advance the secret key can be easily figure out by simply comparing known plaintext attack with corresponding encrypted frame. To solve this problem, binary coin flipping sequence method together with two permutation lists is used. For each 8x8 block a coin is flipped.

If it is a tail, the permutation list 1 (key1) is applied to block. If it is a head, the permutation list 2 (key2) is applied to the block. This method is vulnerable to the ciphertext only attack, because non zero AC coefficients have the tendency to gather in the upper left corner of the block, it would be easy for an adversary to determine which key is used.



**Figure File Encryption Using Sec**

**CHAPTER 3**

**Design and flow process**

**3.1. Evaluation & Selection of Specifications/Features**

It is a lightweight mpeg video encryption which incorporates encryption with MPEG compression in one step [12]. The primary goal of this methodology is to save computation time by taking the advantage of combining MPEG compression and data encryption and at the same time avoid decreasing video compression rate. In this permutation, Huffman codeword list is used as a secret key. During MPEG encoding, the encoder uses the secret key instead of standard Huffman codeword list. Since MPEG compression rate depends on Huffman codeword list, if we use an arbitrarily Huffman codeword list to encode the MPEG video, the compression rate may decrease.

To avoid affecting compression rate, it limits the permutation of Huffman codeword list (secret key) to those codewords which have the same length as the standard Huffman codeword. Second, it seems that not all of permutations of the Huffman codeword list can be used as an encryption keys. This makes key generation difficult since a generated key has to be tested for validity before using.

**Compression Logic based Random Permutation**

The proposed algorithm is Compression logic based video encryption algorithm [13]. Instead of randomly permuting 8x8 coefficients of a single DCT block, the random permutation is applied to a number of permutation groups. Each permutation group contains the DCT coefficients of same frequency from every single block of a frame, regardless of I,P or B frame. Obviously, since each DCT block has 64 coefficients frequencies so that 64 permutation groups can be formed, the proposed algorithm runs random permutations on each of the permutation groups to encrypt a single video frame. After the random permutation the encrypted video data is compressed by standard RLE.

It is also a selective algorithm since only a small number of permutation groups can be encrypted based on the requirements of confidentiality. It is reliable against brute force attacks due to a very large key space. It is secure against DCT vulnerability.

**Correlation Preserving Permutation**

Most encryption algorithms have a randomization effect on the source data, and cannot be effectively applied before compression stage. Using correlation preserving permutation [14] one can perform encryption prior to video encoding. In this scheme, Sorted, as well as “almost sorted” frames are strongly spatial correlated. Such permuted frames are in many instances even more compressible in terms of spatial only coding than the original source frames. When a sorting permutation of previous frame acts on the current frame, it produces what we refer to as an “almost sorted” frame.Transmitting a compressed frame from which the initial permutation can be computed is efficient. Once an initial permutation is transmitted through a secure channel, the sender uses it to “almost sort” the next frame. It is shown that except in rare circumstances, a sorted or “almost sorted” frame can be safely sent through the regular, non secure channel. By calculating a sorting permutation of the received frame, the receiver uses it to recover the next frame. There is no secret key on which a permutation is generated.

This method relies on the sorting permutation of previous frame, and thus, a key is directly dependent on the plaintext. Under a chosen plaintext attack, the adversary can compute the sorting permutation for the chosen frame, but this gives no information about the sorting permutations for the unknown frames. The limited known-plaintext attack is applicable to our method, because the adversary can recover all frames that follow the known frame until the scene changes and key frame is updated.

**Selective Encryption**

In traditional video protection schemes, called fully layered, the whole content is first compressed and then compressed bit stream is entirely encrypted using a standard cipher. This scheme is unsuitable in real time application due to high delay and computation complexity. This section discusses about selective encryption which only encrypt a subset of the data.

The aim of selective encryption is to reduce the amount of data to encrypt while preserving a sufficient level of security.

**Methodology proposed by Meyer and Gadegast**

This methodology is proposed for MPEG videos [15]. This method uses traditional encryption methods RSA or DES in CBC mode to encrypt MPEG video stream. It implements 4 level of security.

(i) Encrypting all stream headers.

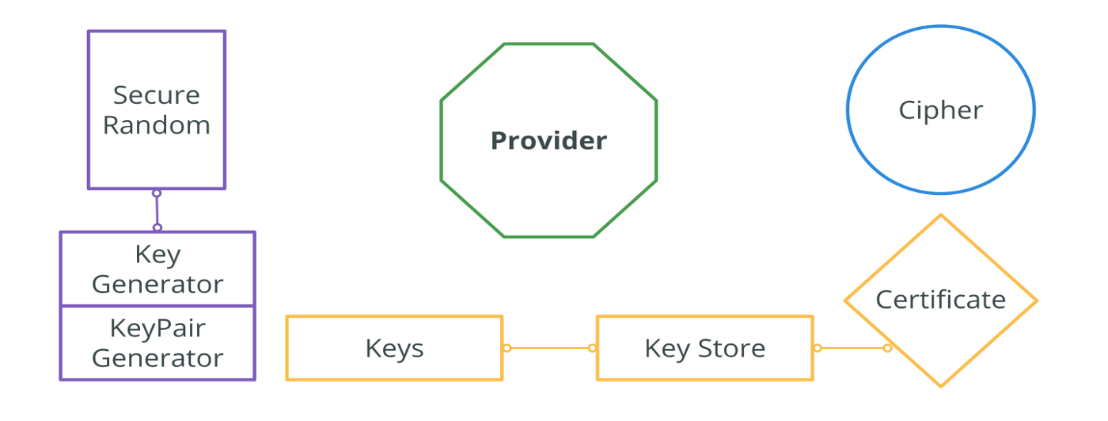
(ii) Encrypting all stream headers and all DC and lower AC coefficients of intracoded blocks.

(iii) Encrypting I-frames and all I-blocks in P- and B frames.

(iv) Encrypting all the bit streams.

The number of I blocks in P or B frames can be of the same order as the number of I blocks in I frames. This reduces considerably the efficiency of the selective encryption scheme [16]. Encryption ratio may vary based on which parameters are encrypted. Encrypting only headers have very less encryption ratio. But encrypting all the bitstreams have 100% encryption ratio. Speed of this methodology again varies based on traditional algorithm in use such as DES or RSA and number of parameters that are encrypted.

Many security levels can be obtained. Encrypting only stream headers is not sufficient since this part is easily predictable. But encrypting all the bit streams can provide high security. Detailed cryptanalysis of this methodology is not defined. A special encoder and decoder are required to read unencrypted SECMPEG stream. The encoder proposed is not MPEG compliant.



Figure

**Methodology proposed by Spanos and Maples**

Aegis mechanism is proposed in [17]; it encrypts intraframes, video stream header and the ISO 32 bits end code of the MPEG stream using DES in CBC mode. Experimental results were conducted by the authors showing the importance of selective encryption in high bitrate video transmission to achieve acceptable end-toend delay.

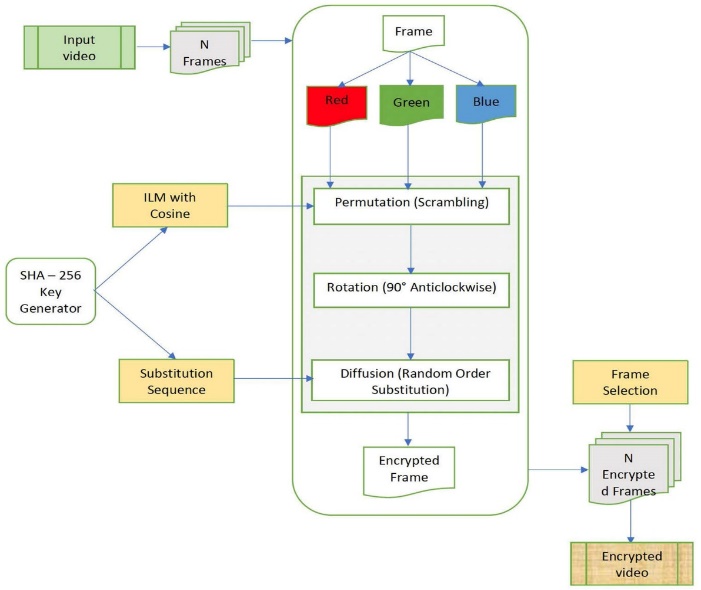
It is also shown that full encryption creates bottleneck in high bitrate distributed video applications. Agi and Gong [18] showed that this algorithm has low security since encrypting of only I-frames offer limited security because of the intercorrelation of frames; some blocks are intracoded in P and B frames. Furthermore, Pand B-frames are highly correlated when they correspond to the same I-frame.

They also underlined that it is unwise to encrypt stream headers since they are predictable and can be broken by plaintext-ciphertext pairs. Alatter and Al-ragib [19], apparently unaware of Agi and Gong work [18], stressed the same security leakage. Encryption is performed after compression, thus no impact is observed on the compression efficiency. The resulting bitstream is not MPEG compliant.

* **Methodology proposed by Shi and Bhargava**.

In [20], the authors proposed video encryption algorithm (VEA) which uses a secret key to randomly change the signs of all DCT coefficients in an MPEG stream. It is fast as it operates on a small portion of original video. It is more efficient than DES algorithm because it only selectively encrypts a small number of bits of the MPEG compressed video and selected bit is only XORed one time with the corresponding bit of the secret key. VEA does not protect from plaintext attack provided the attacker knowns the original video image (plaintext and ciphertext).

In [21], the authors present a new version of VEA reducing computational complexity; it encrypts the sign bits of differential values of DC coefficients of I-frames and sign bits of differential values of motion vectors of Band P-frames. This type of improvement makes the video playback more random and more non viewable. When the sign bits of differential values of motion vectors are changed, the directions of motion vectors change as well.



In addition, the magnitude of motion vectors change, making the whole video very chaotic. Modified VEA encrypt DC coefficients of I frame, and leave AC coefficients of I frames unchanged. Thus it significantly reduces encryption computations. Because DC coefficients of I frames are differentially encoded, changing a few sign bits of differential values of DC coefficients will affect many DC coefficients during MPEG decoding. MPEG’s differential code of DC coefficients and motion vectors increase the difficulty to break MVEA encrypted videos.

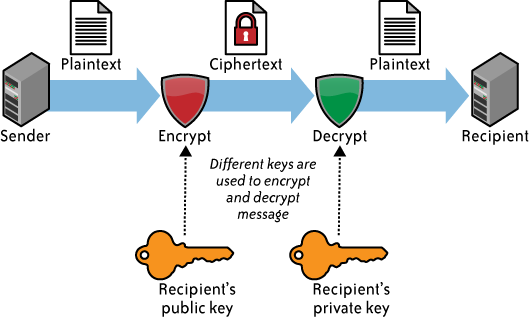
The first version of VEA [21] is only secure if the secret key is used once. Otherwise, knowing one plaintext and the corresponding ciphertext, the secret key can be computed by XORing the DCT sign bits. Both versions of VEA are vulnerable to chosen plaintext attacks; in [21], it is feasible to create a repetitive/periodic pattern and then compute its inverse DCT.

The encryption of the image obtained will allow us to get the key length and even compute the secret key by chosen-plaintext attack.

MHT: The authors propose a method using multiple Huffman coding tables. The input datastream is encoded using multiple Huffman tables. The content of these tables and the order that they are used are kept secret as the key for decryption. In the proposed system, instead of training thousands of Huffman coding tables, it only train and obtained four different Huffman tables.

Then, thousands of different tables can be derived using a technique called Huffman tree mutation. Gillman and Rivest [25] showed that decoding a Huffman coded bit stream without any knowledge about the Huffman coding tables would be very difficult.

However, the basic MHT is vulnerable to known and chosen plaintext attacks as pointed out in [26].



MSI: The arithmetic QM coder is based on an initial state index; the idea is to select 4 published initial state indices and to use them in a random but secret order. Unlike Huffman coding with a fixed and pre defined Huffman tree, the QM coder dynamically adjusts the underlying statistical model to a sequence of received binary symbols. It is very difficult to decode the bitstream without the knowledge of the state index used to initialize the MQcoder. A little effect on compression efficiency is observed. This is due to multiple initializations of the QM coder due to initial state index changing.

**3.2. Design Constraints**

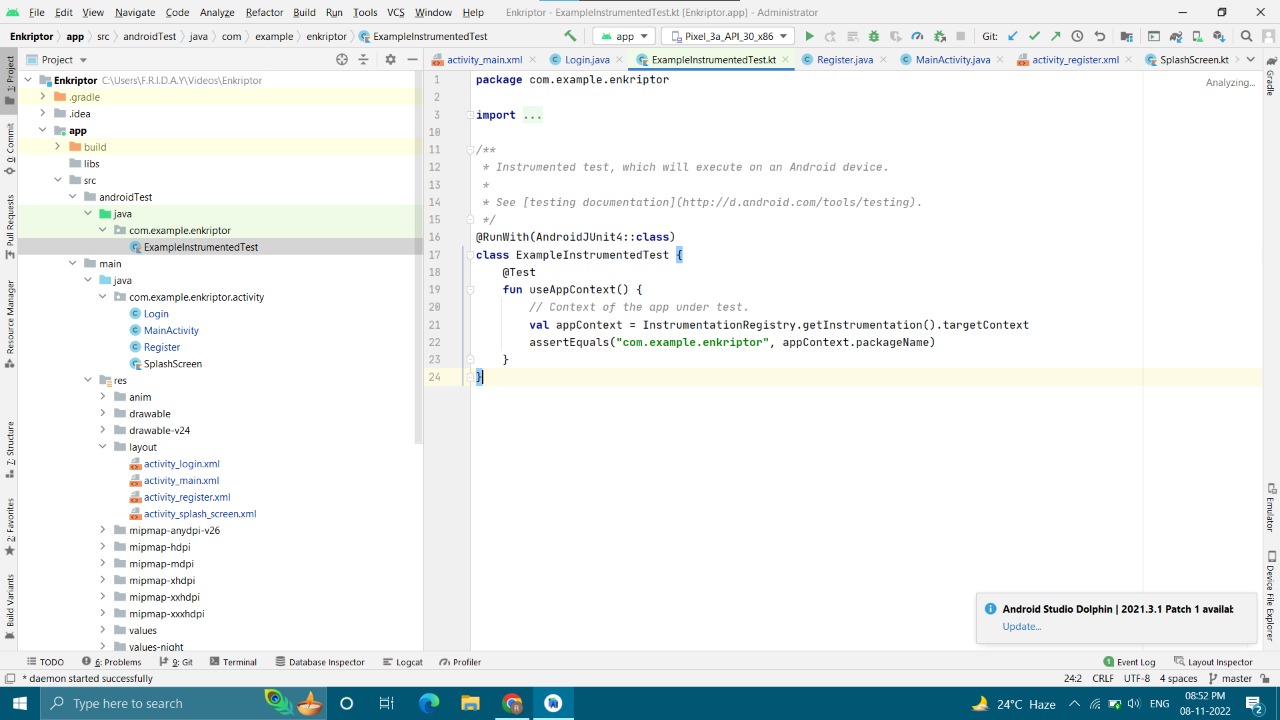


Fig 2

**Unit Test Activity Java File**

Activities serve as containers for every user interaction within your app, so it's important to test how your app's activities behave during device-level events, such as the following:

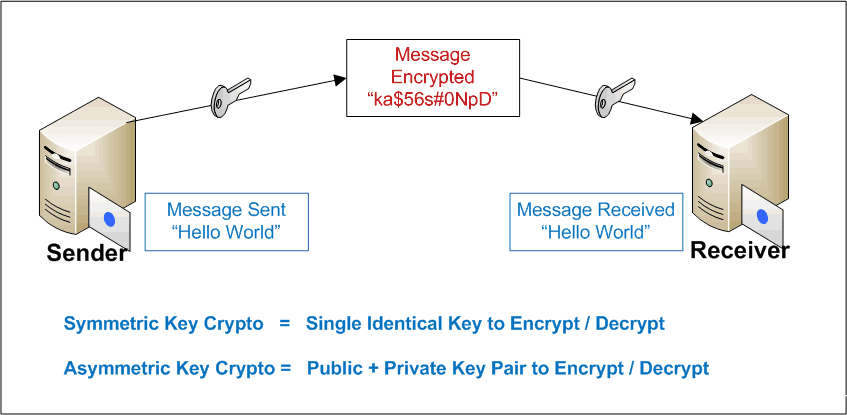
* Another app, such as the device's phone app, interrupts your app's activity.
* The system destroys and recreates your activity.
* The user places your activity in a new windowing environment, such as picture-in-picture (PIP) or multi-window.
* In particular, it's important to ensure that your activity behaves correctly in response to the events described in Understanding the life cycle.

**A . Attacker Model**

Attacker models are a fundamental part of research on security of any system. For different application scenarios, suitable attacker models have to be chosen to allow comprehensive coverage of possible attacks.

We consider Cyber-Physical Systems (CPS), that typically consist of networked embedded systems which are used to sense, actuate, and control physical processes.

The attacker can either initiate a request or can respond to a request. In either case, the attacker can encrypt a real video or create an arbitrary collection of bytes to attempt to defeat the system. We assume that the cryptosystem is strong enough to prevent an attacker from decrypting the data [8].



**Attacker Model**

**B. System Design**

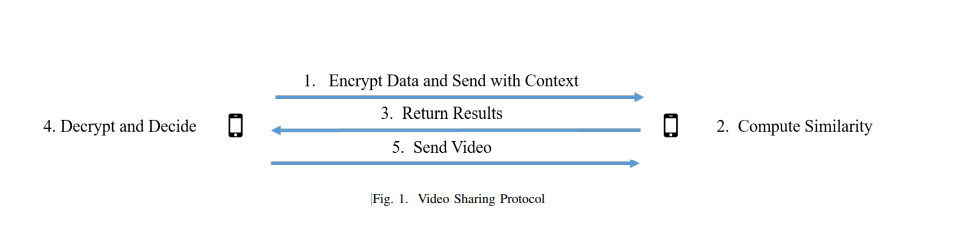
Our system leverages two main technologies to enable video sharing. The first is Similarity of Simultaneous Observation (SSO) . The algorithms presented in [9] enable a user to determine with high accuracy if two videos are of the same scene at the same time.

There is no accurate and standard answer to the design problems. You may have different conversations with different interviewers for the same question.

Due to the open-ended nature of this round, not just junior and mid-level developers but also experienced developers feel uncomfortable in this round.

The second is fully homomorphic encryption (FHE) [10] which enables us to execute algorithms on encrypted data. A cryptosystem is said to be fully homomorphic if it is homomorphic for all algorithms.

Due to complexities of developing efficient algorithms under a FHE cryptosystem, we have had to modify the original SSO algorithm so that it could be used in our system. Since



**Video Sharing Protocol**

FHE computational libraries have not reached the maturity of traditional mathematical libraries, many functions that developers take for granted are not natively available under an FHE cryptosystem. As a result, we had to use a combination of pre-processing, function approximations through Taylor expansion, and the use of different distance measures in order to achieve similar results the original SSO paper.

The original SSO system used Kullback-Liebler Divergence (KLD), Jensen-Shannon Divergence (JSD), Dynamic Time Warping (DTW), and Pearson Correlation Coefficient (PCC) as the similarity measures. Due to poor classification performance, we eliminated DTW and PCC from this system. Additionally, the computation required by JSD required expensive approximations when done as a FHE algorithm that were prone to producing significant errors in certain regions of the input space, so we did not use it. We were able to approximate KLD under FHE through the use of precomputing and encrypting several values that would later be used under the FHE scheme.

We also identified two additional similarity measurements, Bhattacharyya coefficient and Cramer distance that we were able to implement efficiently under the FHE cryptosystem all with mean approximation errors of less than 0.62% of the mean difference between a similar video and a different video.

Figure 1 demonstrates the protocol used to determine if the two videos are of the same scene once the user on the right has initiated a request. The process can be summarized as follows:

**3.4. Design Flow**

1. **Encrypt Data and Send with Context.**

The user that is making the decision whether or not to share the video preprocesses the video and encrypts it. The encrypted data and the context is sent to the other user that is trying to prove they were co-present over a mutually authenticated TLS connection.

**Compute Similarity.**

The other user computes all of the requisite similarity scores. Note that the other user cannot directly see what effect the values they use as inputs into the algorithms have. They can only see if they are sent the file or not.

**Return Results.**

The encrypted results are returned to the decision-making user.

**Decrypt and Decide.**

The decision-making user decrypts the results and provides them as inputs into the classification model.

**Send Video.**

If the decision-making user is satisfied with the results, they send the video to the other participant.

**3.5. Design selection**



**Outer Design**

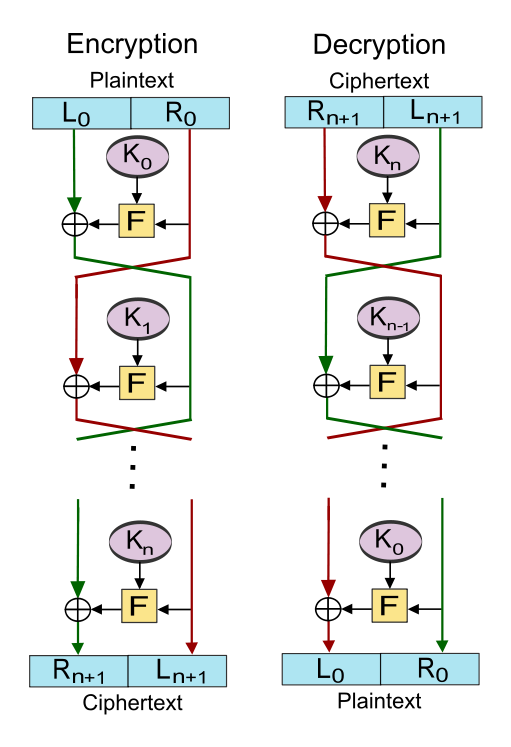
## 

**Themes.xml**

Themes.xml file describes the theme of the android app which includes color scheme during night mode and normal mode.

XML stands for eXtensible Markup Language, which is a way of describing data using a text-based document.

Because XML is extensible and very flexible, it's used for many different things, including defining the UI layout of Android apps.



**Encryption**

**3.6. Implementation plan/methodology**

have implemented our Proof of Presence Video Sharing system (PoP-Share) for mobile devices. Our mobile phone implementation is designed to run as an Android App. The app is built for Android 9.0 and runs on 64 bit CPUs. The app uses SEAL 3.3 [11] built using the Android NDK, so this implementation uses the CKKS implementation of fully homomorphic encryption.

We have implemented all of the FHE functionality in native C++ with a JNI wrapper to be accessed through the Android app. We currently are using a native Android GUI, but will be implementing the App with the Kivy GUI in the near future so the mobile phone user experience will match that of the PC implementation user experience.

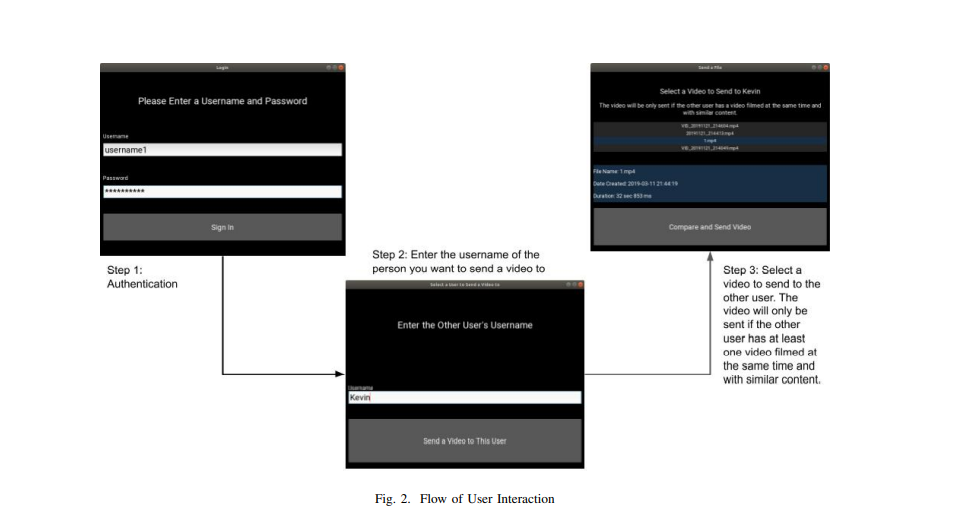
SEAL also runs on iOS, but we have not yet ported PoP-Share to that platform.

1. **Performance**

1) Timing: On an Android phone, the preprocessing time, which includes precomputation and encryption of all the data takes between 4 and 8 seconds on a Pixel 2 mobile phone to compare 60 seconds of video. While this is a large amount of time for pre-processing on the mobile phone, this is a process does not have to be run every time.

1. **Classification**

As noted in table I, our system performs similarly to the original SSO system in terms of classification. We optimized our classifier for precision rather than F1 score because we considered the cost of a false positive to be much worse than a false negative.



# Fig 2. Flow of User Interaction

As noted in table I, our system performs similarly to the original SSO system in terms of classification. We optimized our classifier for precision rather than F1 score because we considered the cost of a false positive to be much worse than a false negative.

# CHAPTER 4

**RESULTS ANALYSIS AND VALIDATION**

4.1. Implementation of solution

Use modern tools in:

* Analysis

Videos are transferred through various types of computer network. To secure video communication different encryption methodologies are used.

Due to huge size of digital videos they are generally transmitted in compressed formats such as MPEG 1/2/4[4][5][7], H.263/ H.264/AVC.

Various encryption algorithms have been proposed in literature. This section presents detail description of these algorithms.

* Design drawings/schematics/ solid models

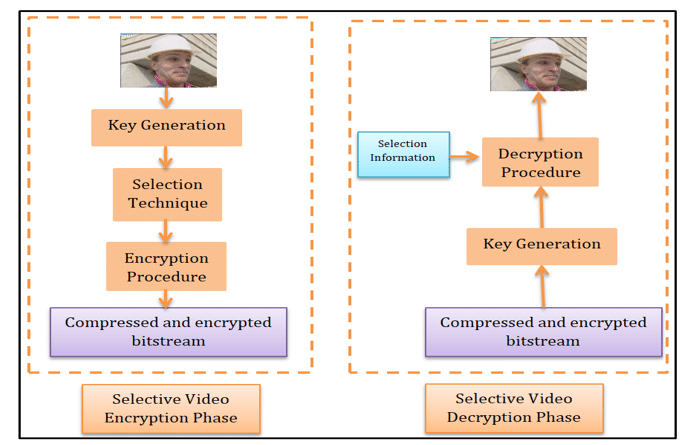


Fig 3. Key Generation Technique

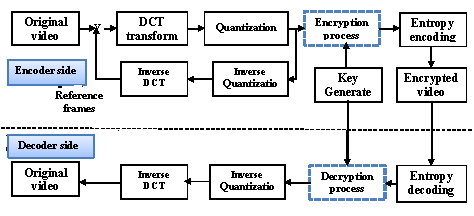


Fig 4. Flow diagram

* report preparation,

We will bring several devices such as laptops and mobile phones that have our software preloaded on them that we will use to demonstrate the video sharing software. In addition, we will also provide a QR code that will link to an APK that Android users will be able to install on their mobile phones so that they can also participate in the demo with their own devices and try it in other areas besides in front of our poster. We intend to connect to the venue’s Wi-Fi to perform the demonstration

* Testing/characterization/interpretation/data validation.

Eight video files have been tested on smart phones using fully encryption. 256 points must be generated on a curve and each input character is converted to byte code according to the standard ASCII code table for video representation.

## Unit Testing

The automated testing method laid out above where the input to a program is modified is quite convenient, but limited nonetheless. Testing larger programs in this way is challenging. One solution to this is unit testing, where small parts of the program are tested in isolation.

Unit testing refers to the testing of individual components in the source code, such as classes and their provided methods.

The writing of tests reveals whether each class and method observes or deviates from the guideline of each method and class having a single, clear responsibility. The more responsibility the method has, the more complex the test.

## 

## UnitTest.java

Unit testing is done to ensure that developers write high-quality and errorless code. It is advised to write Unit tests before writing the actual app, you will write tests beforehand and the actual code will have to adhere to the design guidelines laid out by the test. In this article, we are using JUnit to test our code. JUnit is a “Unit Testing” framework for Java Applications which is already included by default in android studio.

It is an automation framework for Unit as well as UI Testing. It contains annotations such as @Test, @Before, @After, etc. Here we will be using only @Test annotation to keep the article easy to understand.

Note that we are going to implement this project using the Kotlin language.

Android platform with version Jelly Bean

(4.1) is used for ECC implementation. Table 2 presents the video files and the time required for encryption and decryption and with charts.

The measurements in the table refer that the implementation of EC in smart phones contributed positively in solving the limitations of video phones security.

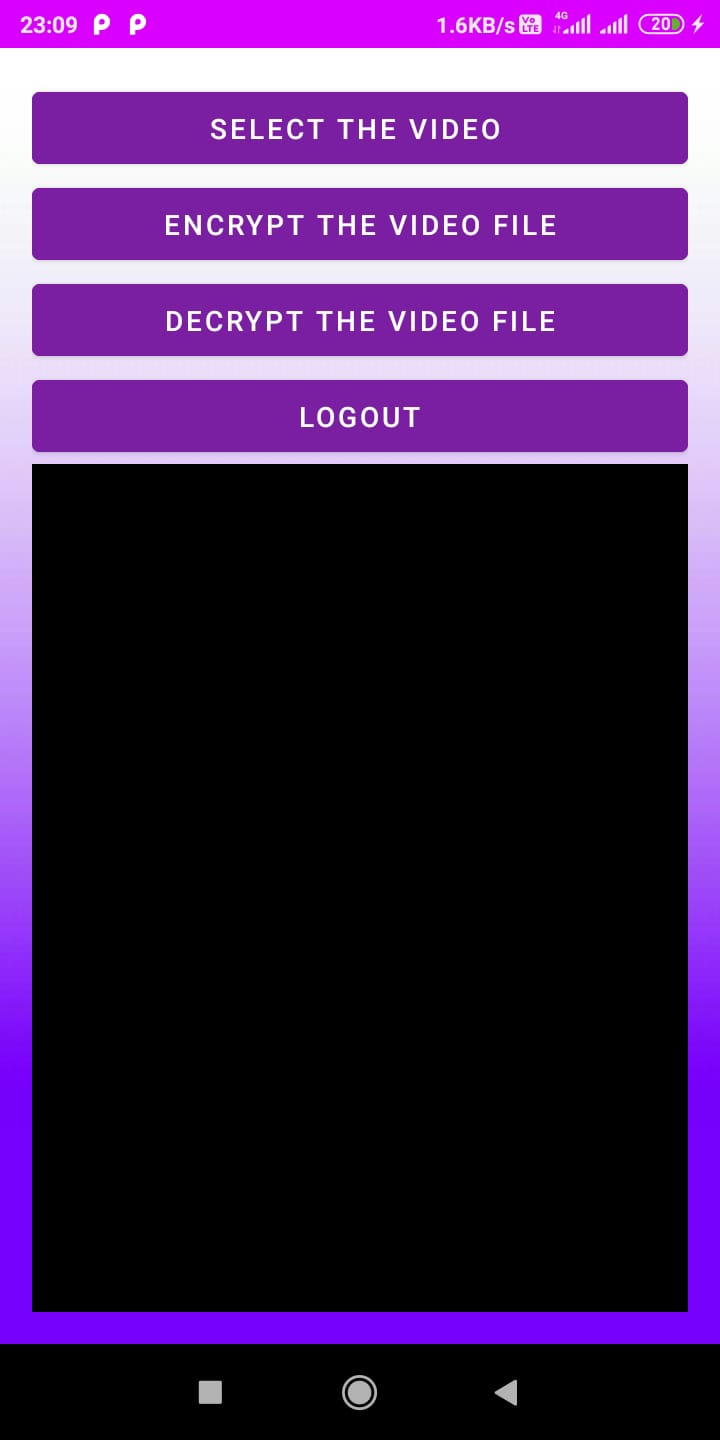
****

Fig Main activity

# Introduction to activities

The Activity class is a crucial component of an Android app, and the way activities are launched and put together is a fundamental part of the platform's application model. Unlike programming paradigms in which apps are launched with a main() method, the Android system initiates code in an Actiivity instance by invoking specific callback methods that correspond to specific stages of its lifecycle.

The Activity  class is designed to facilitate this paradigm. When one app invokes another, the calling app invokes an activity in the other app, rather than the app as an atomic whole. In this way, the activity serves as the entry point for an app's interaction with the user. You implement an activity as a subclass of the Activity class.

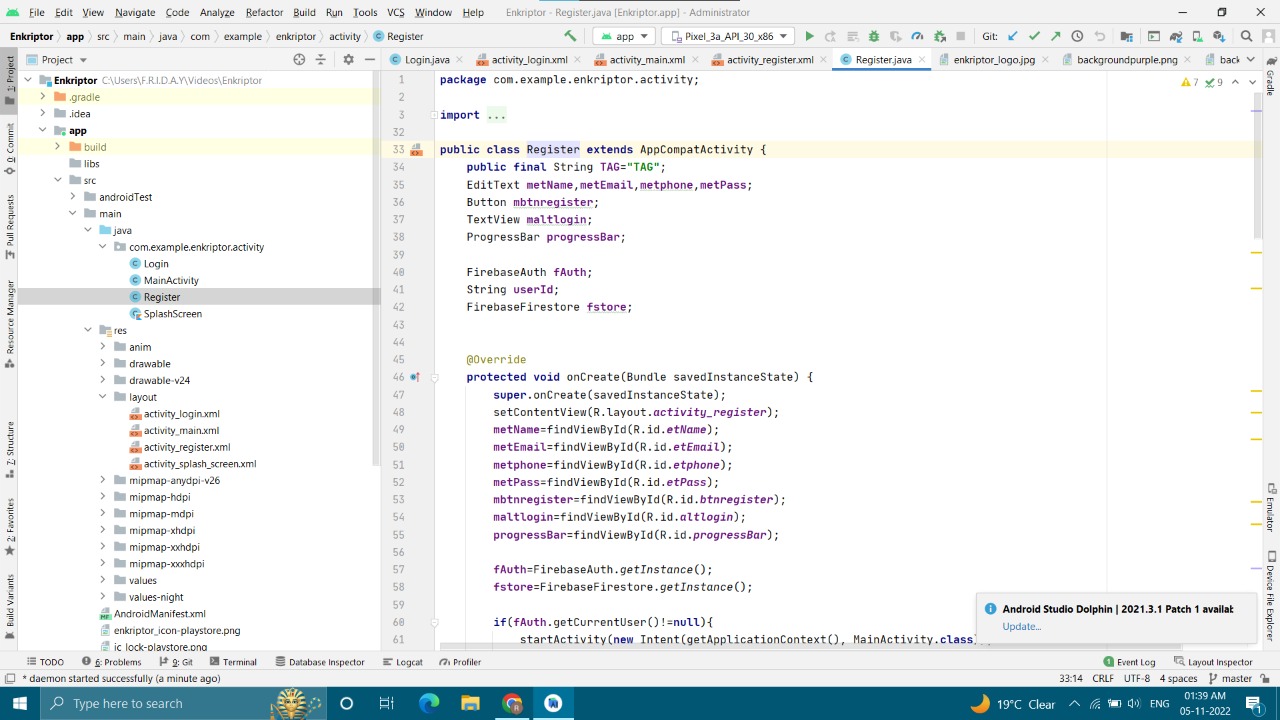


Fig Register Activity Java File

**Splash Screen Java File**

Splash Screen is most commonly the first startup screen which appears when App is opened. In other words, it is a simple constant screen for a fixed amount of time which is used to display the company logo, name, advertising content etc. **Splash Screen Implementation Method In Android:**

**Method 1 of implementing Splash Screen**:

Create a thread and set time to sleep after that redirect to main app screen.

**Method 2 of Implementing Splash Screen:**

Set time to handler and call Handler().postDelayed, it will call run method of runnable after set time and redirect to main app screen.

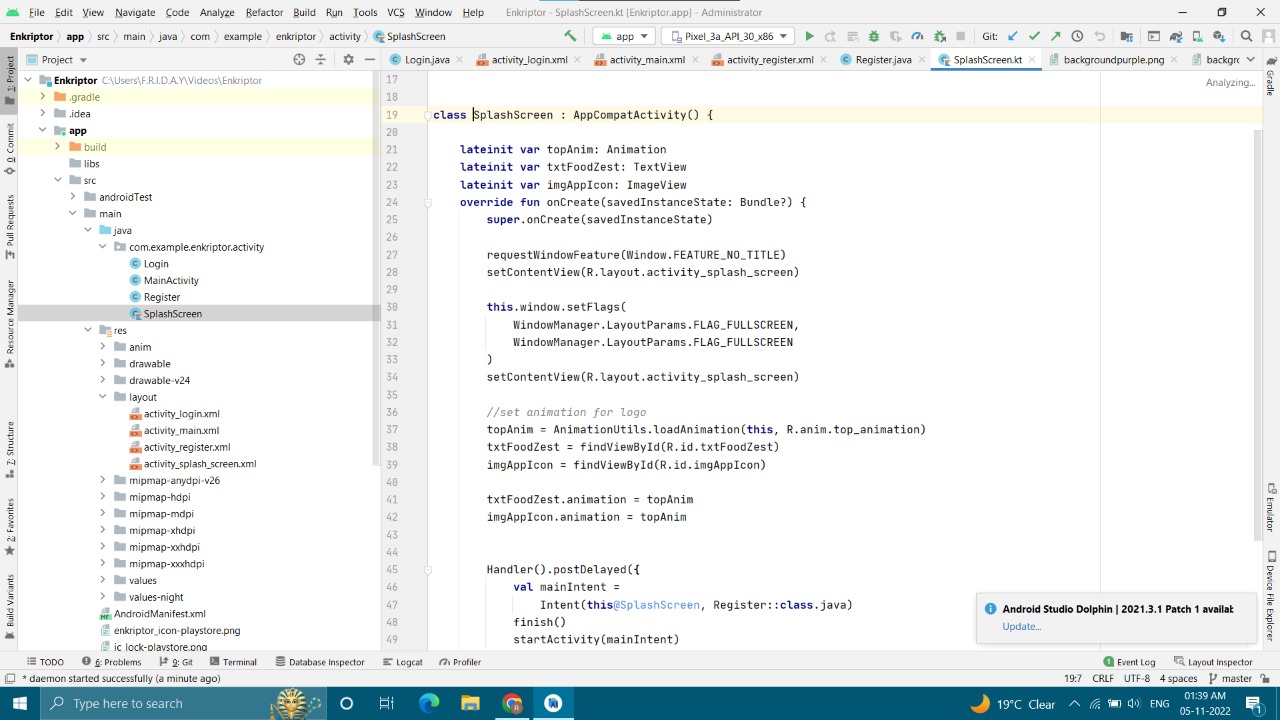


Fig Splash Screen Java File

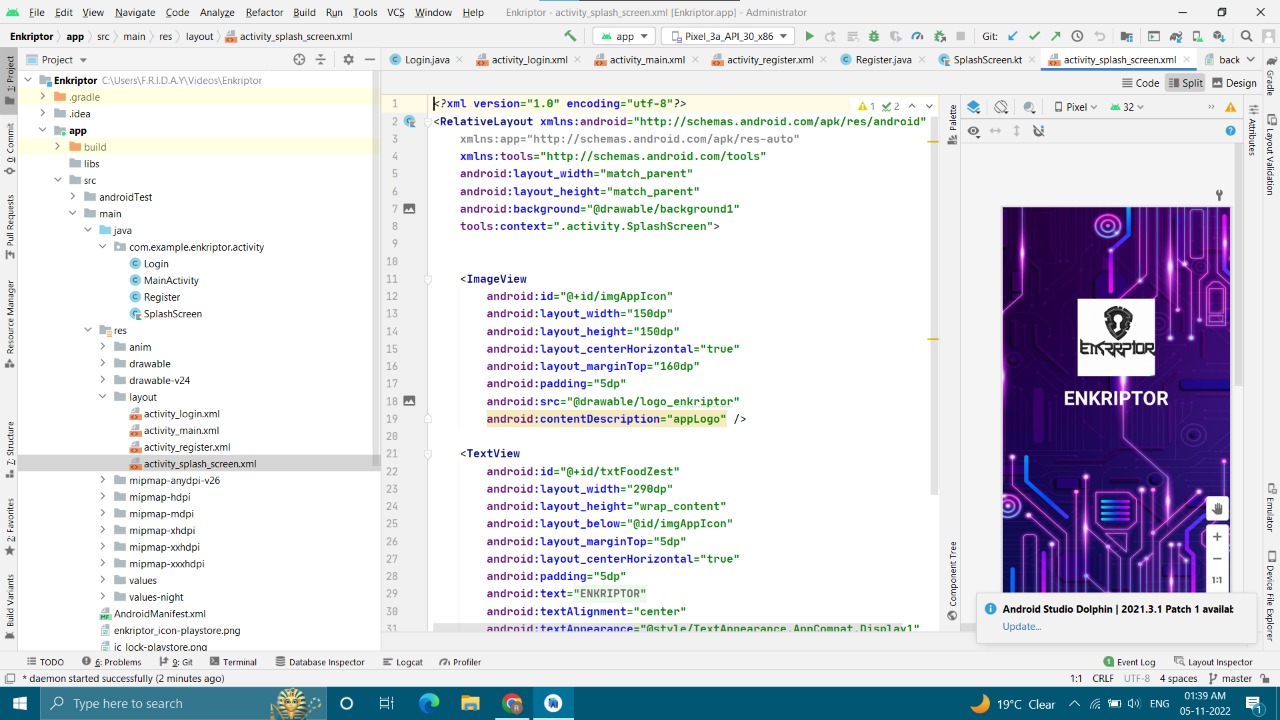


Fig Splash Screen Activity XML File

Splash Screen Activity is the activity which appears when we click the app icon it has the animation of logo coming from top to bottom. And the code of the animation is provided in the top animation xml file.

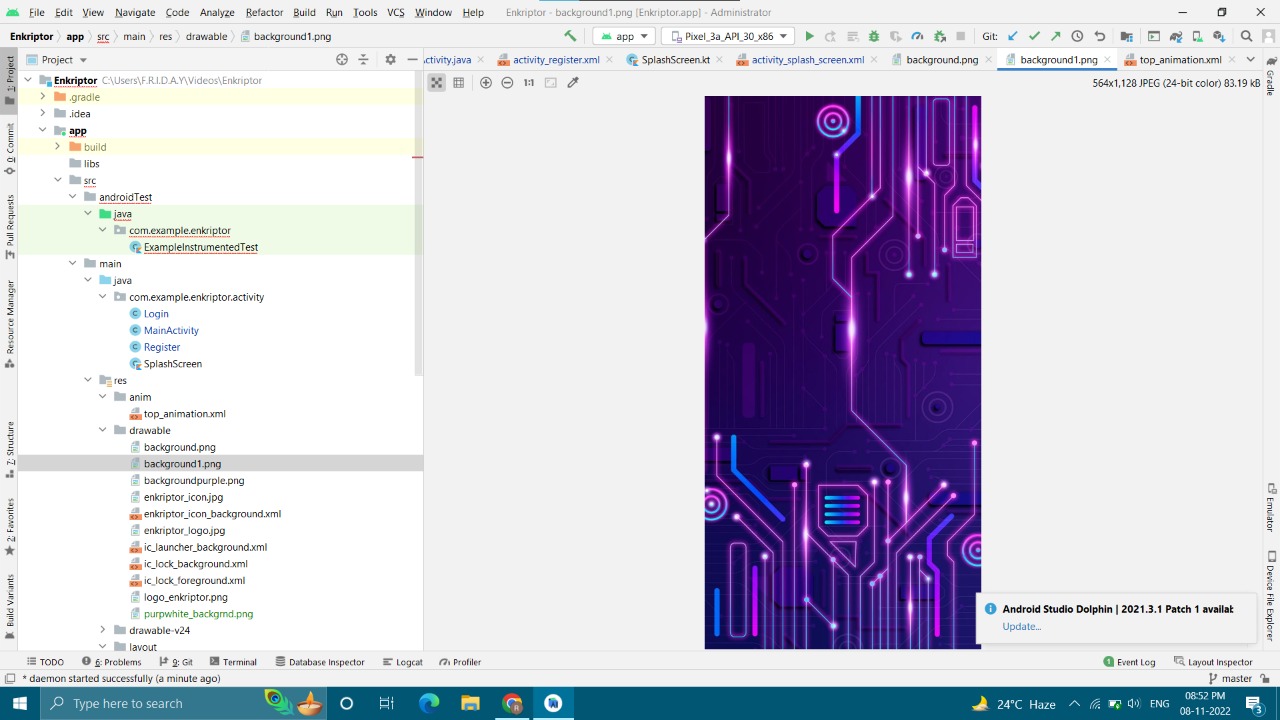


Fig Splash Screen Background



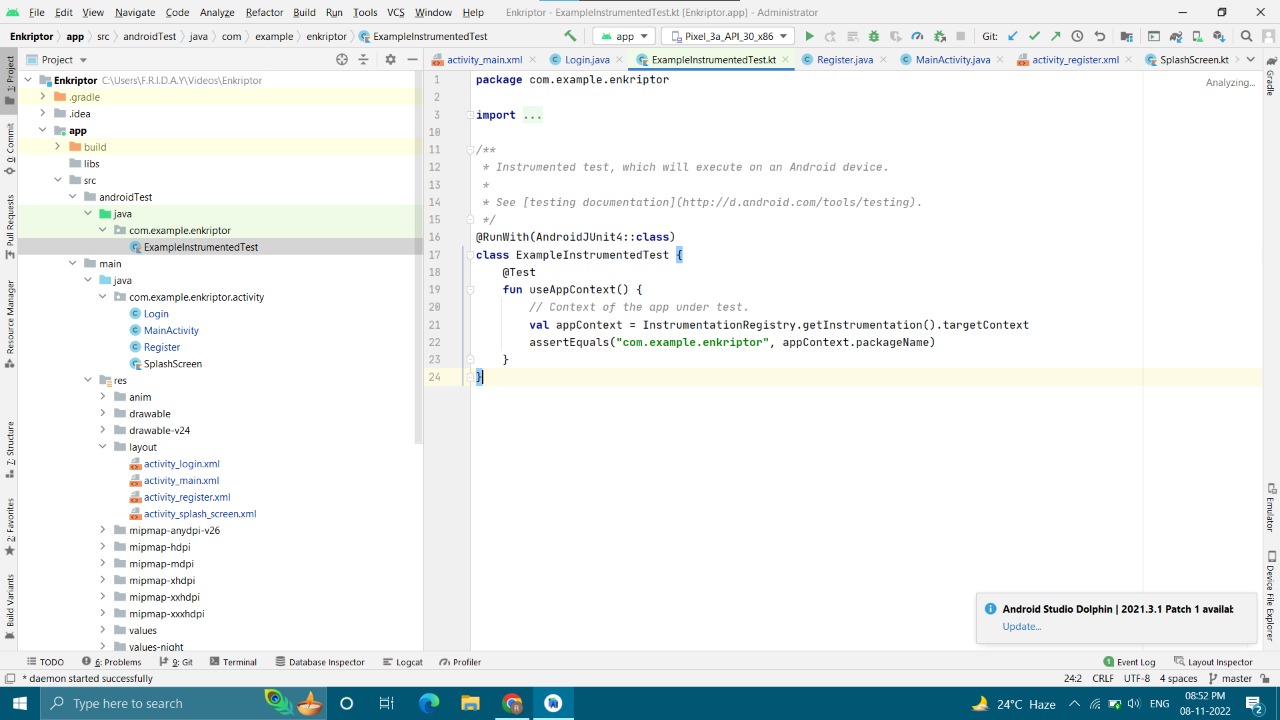
Fig Animation XML File

# Build instrumented tests

Instrumented tests run on Android devices, whether physical or emulated. As such, they can take advantage of the Android framework APIs. Instrumented tests therefore provide more fidelity than local tests, though they run much more slowly. We recommend using instrumented tests only in cases where you must test against the behavior of a real device. AndroidX Test provides several libraries that make it easier to write instrumented tests when necessary.

In your Android Studio project, you store the source files for instrumented tests in module-name/src/androidTest/java/. This directory already exists when you create a new project and contains an example instrumented test.

Before you begin, you should add AndroidX Test APIs, which allow you to quickly build and run instrumented test code for your apps.



Unit Test Activity Java File

**Android Animation**

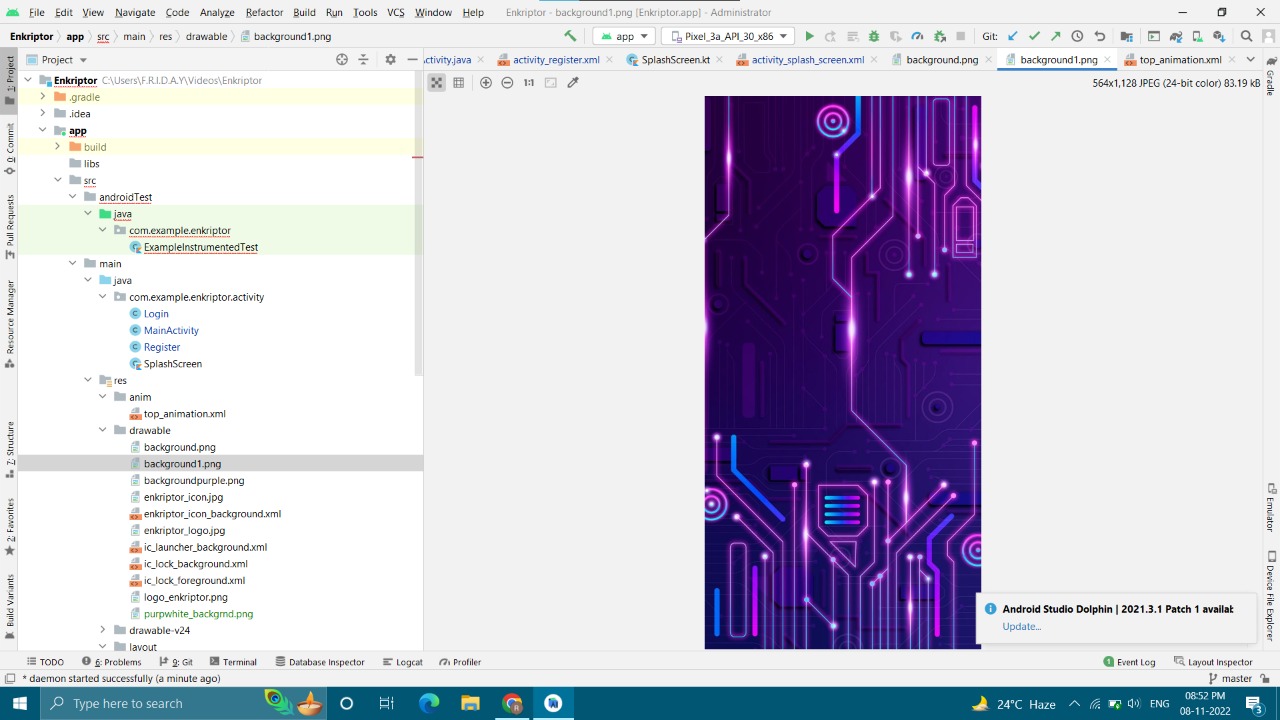
Animation in android apps is the process of creating motion and shape change. Android Animation is used to give the UI a rich look and feel. Animations in android apps can be performed through XML or android code. In this android animation tutorial we’ll go with XML codes for adding animations into our application.

To apply Animations to our Application sometimes we need to make an anim folder in Android Studio to store animation file under the resource folder of our application. Basically there are two options for resource types.

You can set any of these two given below:

animator/ -XML files that define property animations.

anim/ – XML files that define tween animations



Splash Screen Backround png image

# AndroidManifest.xml file in android

The **AndroidManifest.xml file** contains information of your package*,* including components of the application such as activities, services, broadcast receivers, content providers etc.

It performs some other tasks also:

It is responsible to protect the application to access any protected parts by providing the permissions.

It also declares the android api that the application is going to use.

It lists the instrumentation classes. The instrumentation classes provides profiling and other informations. These informations are removed just before the application is published etc.

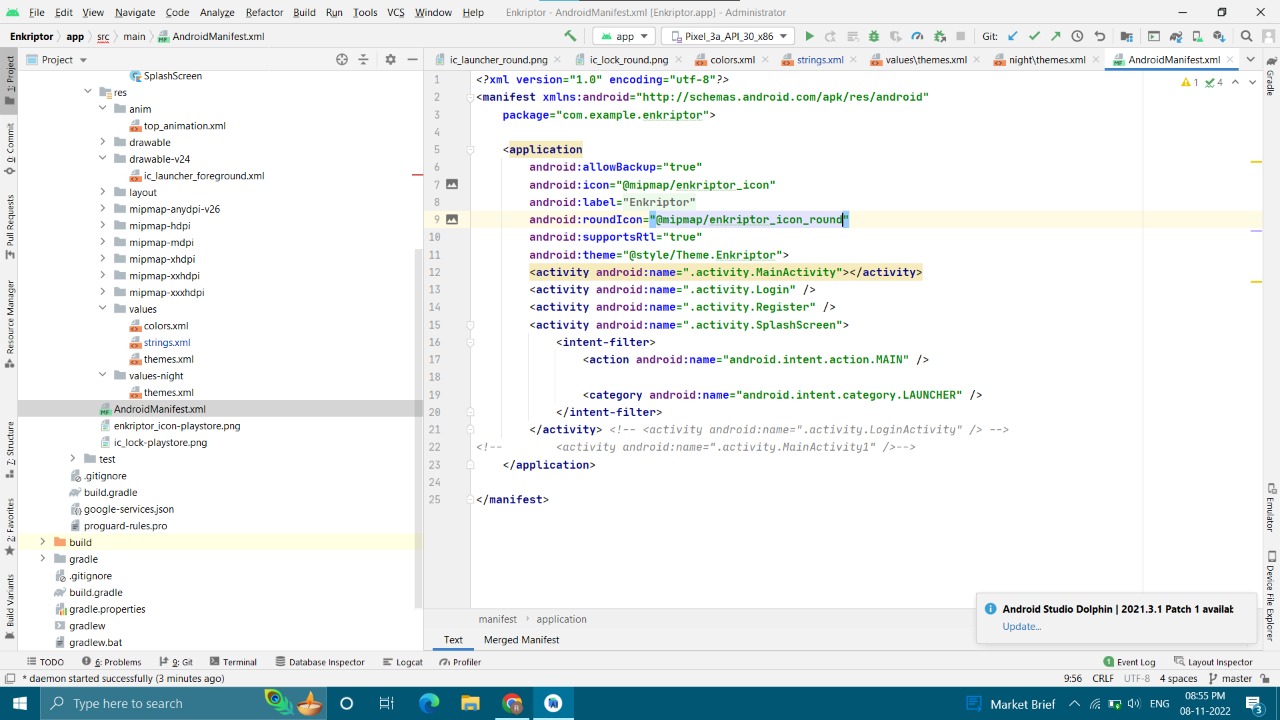


Fig Android Manifest XML File

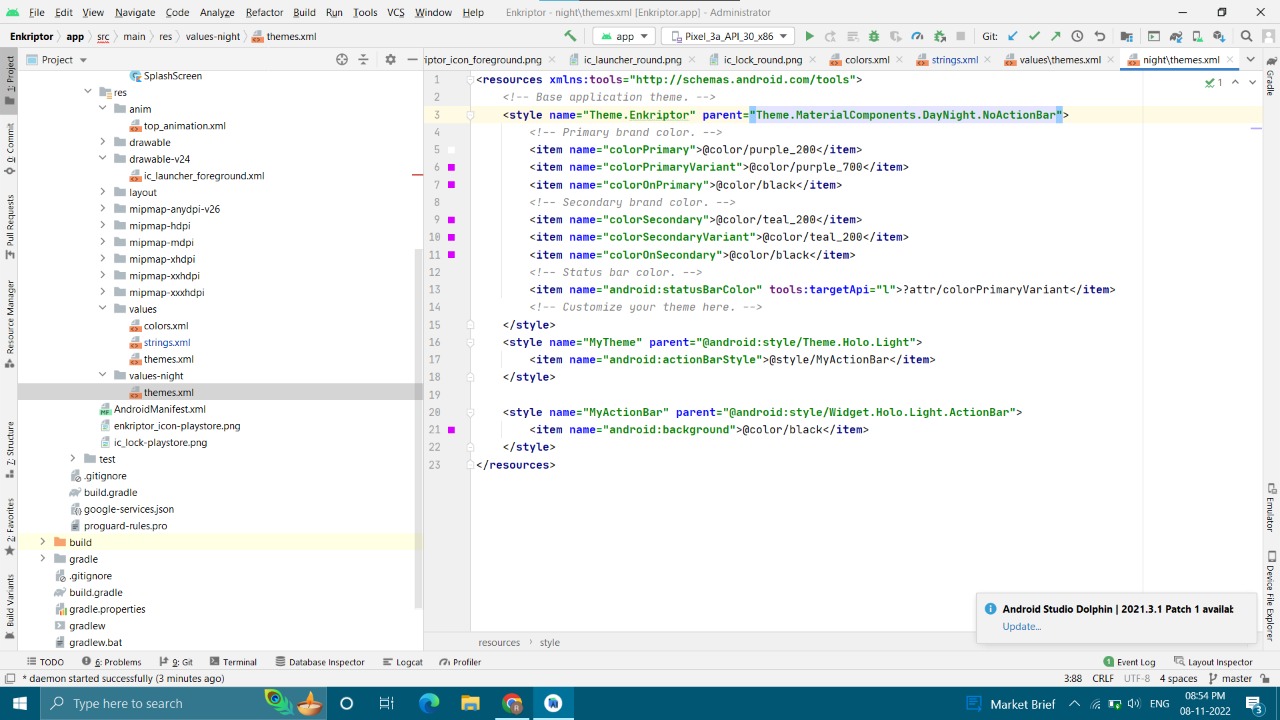


Fig Themes.XML File

The Android styling system offers a powerful way to specify your app’s visual design, but it can be easy to misuse. Proper use of it can make themes and styles easier to maintain, make branding updates less scary and make it straightforward to support dark modes. Both themes and styles use the same <style> syntax but serve very different purposes. You can think of both as key-value stores where the keys are attributes and the values are resources. A style is a collection of view attribute values.

You can think of a style as a Map<**view** attribute, resource>. That is the keys are all view attributes i.e. attributes that a widget declares and you might set in a layout file. Styles are specific to a single type of widget because different widgets support different sets of attributes:

A style applied to a view **only** applies to that view, not to any of its children.

**For example**, if you have a ViewGroup with three buttons, setting the InlineAction style on the ViewGroup will not apply that style to the buttons. The values provided by the style are combined with those set directly in the layout.

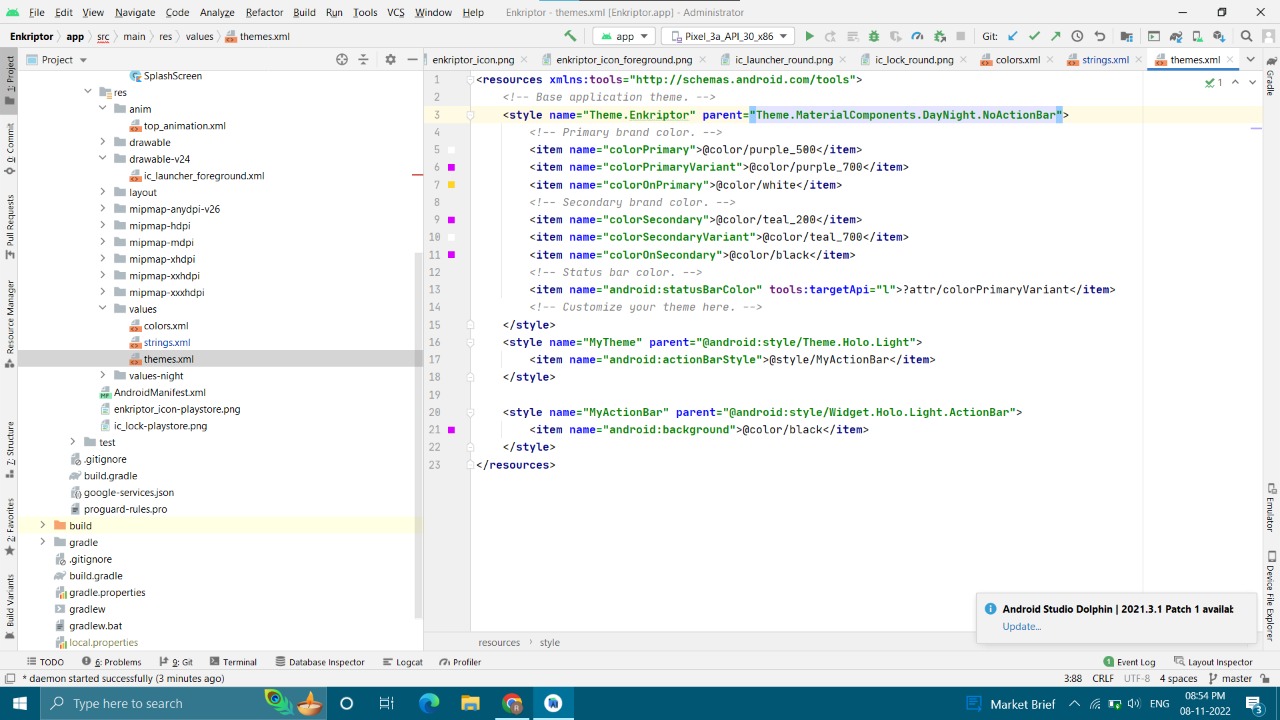


Fig Night Theme.XML File

A theme is a collection of named resources which can be referenced later by styles, layouts etc. They provide semantic names to Android resources so you can refer to them later e.g. colorPrimary is a semantic name for a given color:

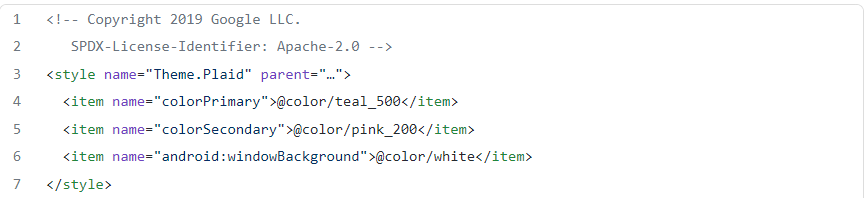


Fig colorschema in Themes.xml

These named resources are known as theme attributes, so a theme is Map<**theme** attribute, resource>. Theme attributes are different from view attributes because they’re not properties specific to an individual view type but semantically named pointers to values which are applicable more broadly in an app. A theme provides concrete values for these named resources. In the example above the colorPrimary attribute specifies that the primary color for this theme is teal. By abstracting the resource with a theme, we can provide different concrete values (such as colorPrimary=orange) in different themes.

**String.xml File**

String. xml file contains all the strings which will be used frequently in Android project. String. xml file present in the values folder which is sub folder of res folder in project structure.In Android Studio, we have many Views such as TextView, Button, EditText, CheckBox, RadioButton etc.

A string resource provides text strings for your application with optional text styling and formatting. There are three types of resources that can provide your application with strings:

**String**

XML resource that provides a single string.

**String Array**

XML resource that provides an array of strings.

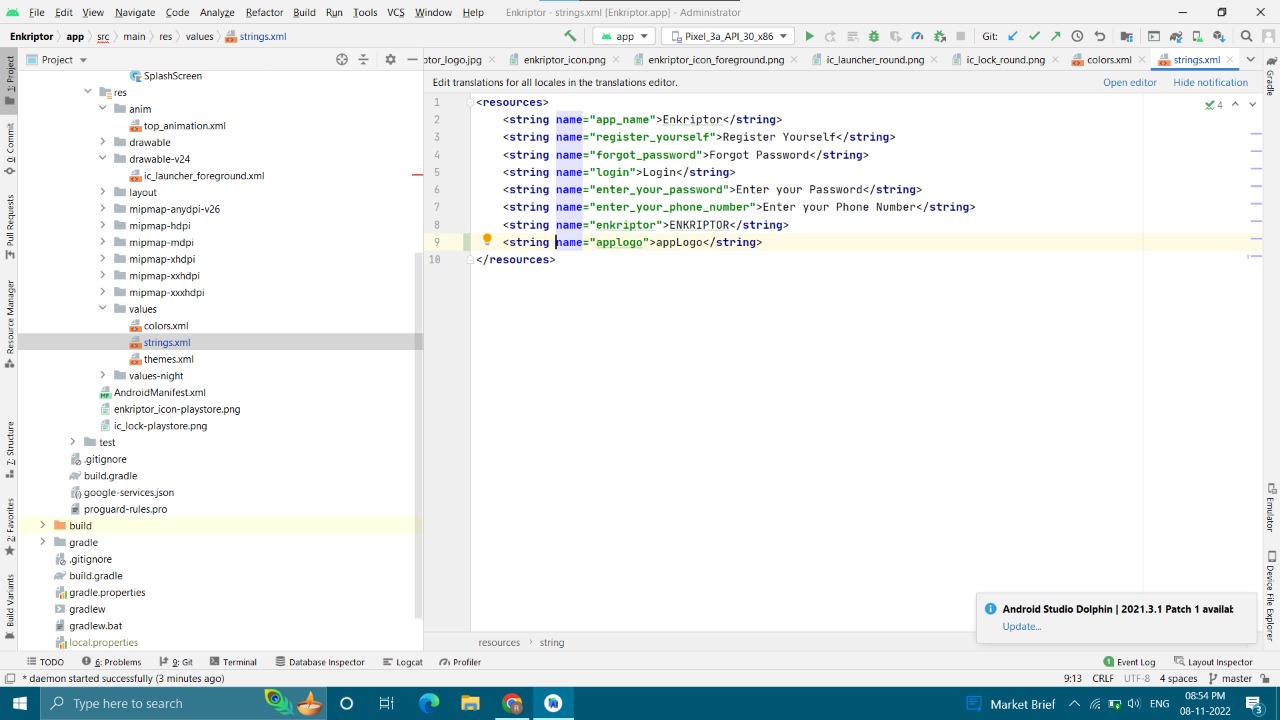


Fig String.xml File

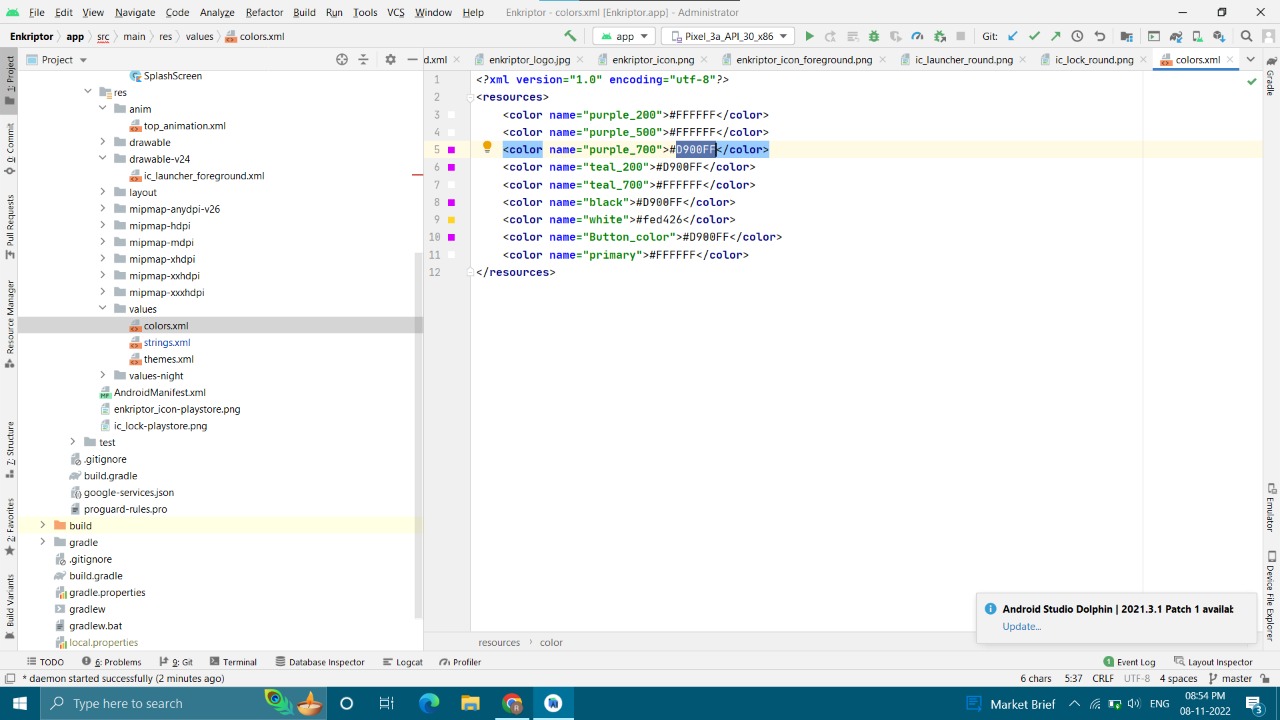


Fig Colors.XML File

Styles and themes on Android allow you to separate the details of your app design from the UI structure and behavior, similar to stylesheets in web design.

A style is a collection of attributes that specify the appearance for a single View. A style can application for the Android based mobile devices. Although an important and rich variety of video encryption algorithms have been used, most of the algorithms defined are not completely secured. Thisapplication allows users to send file or data in the video format to other android devices in a secure network. In this way, application can still access the data without reaching for user’s sensitive information. This application can be useful in many real time events and applications for an enhanced user support. It is observed that the decryption process takes more time relatively as compared with the encryption process, which is acceptable because of the nature of the algorithm besides this the algorithms are implemented in limited resources. The main advantage of this system is achieving the protection for data of video in android devices such as confidentiality for the secure end-to-end user communication and for simple user interface so that it is easy for users to interact.

# Gradle Build

The **Gradle build** is a process of creating a Gradle project. When we run a gradle command, it will look for a file called **build.gradle** in the current directory. This file is also called **the Gradle build script**. The build configuration, tasks, and plugins are described in this file. The build script describes a project and its tasks.

The build.gradle file is build script of a Gradle project. All the tasks and plugins are defined in this file. When we run a gradle command, it looks for a file called build.gradle in the current directory. Although we have called it a build script, strictly, it is a build configuration script. The build script defines a project and its tasks.

## 

## Fig Build.gradle File

## 

## Fig Dependencies

## Dependencies

Gradle build script describes a process of building projects. Most of the projects are not self-contained. They need some files to compile and test the source files. For example, to use Hibernate, we must include some Hibernate JARs in the classpath. Gradle uses some unique script to manage the dependencies, which needs to be downloaded.

The dependencies are used to assist a task, such as required JAR files of the project and external JARs. Every dependency is applied to a specified scope. For example, dependencies are used to compile the source code, and some will be available at runtime.

Gradle signifies the scope of a dependency with the help of a Configuration, and a unique name can recognize every configuration. Most of the Gradle plugins support pre-defined configuration for the projects.

Gradle considers the outcomes of building and publishing the projects. Publishing is based on the task that we define. It can copy the files to a local directory or upload them to a remote Maven or lvy repository. We can use these files from another project in the same multi-project build. The process of publishing a task is called publication.

## 

## Fig Google Services.json

## JSON Google Services in Android

## What it does.

So this plugin and JSON file are not essential to running or publishing your app, it is just a quickstart helper to generate some basic android-resource files for easier integration of specific Google API features.

The google-services.json file

It is generally placed in the app/ directory (at the root of the Android Studio app module).

As of version 2.2.0 the plugin supports build type and product flav specific. JSON files. All of the following directory structures are valid.

## Gradle SW:

## Gradle makes it easy to build common types of project — say Java libraries — by adding a layer of conventions and prebuilt functionality through plugins. You can even create and publish custom plugins to encapsulate your own conventions and build functionality.

## Gradle Wrapper allows you to execute the Gradle builds on machines, where Gradle is not

## installed. This is useful for continuous integration of servers.

## Following are the Gradle Wrappers used in the development:

## 

## Fig Gradlesw

## 

## Fig Gradlesw

## 

## Fig Gradlesw

The complete Gradle API is designed using Groovy language. This is an advantage of an internal DSL over XML. Gradle is a general purpose build tool and its main focus is Java projects.

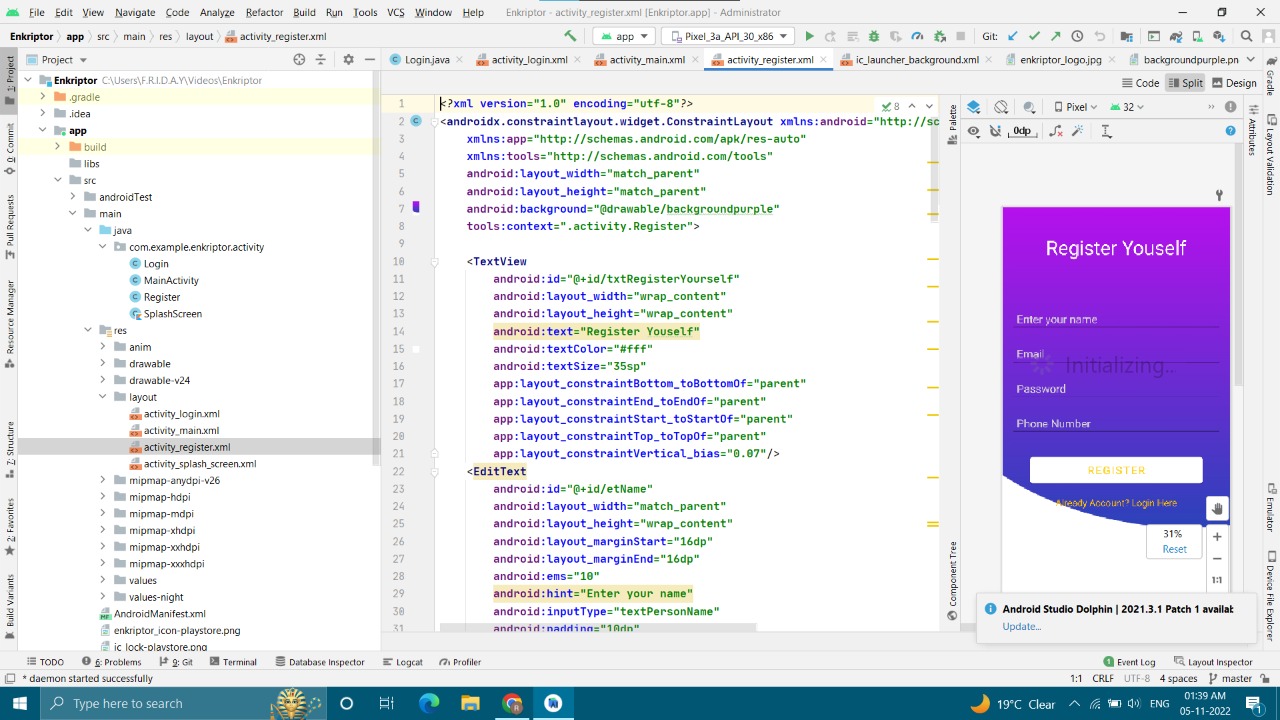
In such projects, the team members will be very familiar with Java and it is better that a build should be as transparent as possible to all the team members.

Languages like Python, Groovy or Ruby are better for build framework. The reason for choosing Groovy is, because, it offers by far the greatest transparency for people using Java. The base syntax of Groovy is same as Java and Groovy provides much more benefits for its users.

**Register Activity**

Regsiter activity consists of total seven viewgroups two textfields four edittexts and one button.

Following is the xml file showing that.



**Fig 4. Register Activity XML File**

**Chapter 5**

**Conclusion And Future Work**

**5.1 Conclusion**

Although an important and rich variety of video encryption algorithms have been proposed in literature, most of the algorithms are not secure against cryptanalysis attack. Naïve algorithm provides highest level of security but it is very slow in nature and cannot be used in real time. Permutation based algorithms are generally faster but they do not provide sufficient level of security. Selective encryption algorithms reduce computational complexity by selecting only a minimal set of data to encrypt but their security and speed level generally vary based on which and how many parameters they encrypt.

Perceptual encryption algorithms are suitable for application like pay per view TV, video on demand where potential users like to see low quality video before buying them. So, these algorithms are not suitable for applications which demand high security. It is difficult for a single algorithm to satisfy all performance parameters. So, selection of encryption algorithm always depends on requirements of application in use.

By looking at table we can conclude that it is a challenge for researchers to design an encryption algorithm which maintains tradeoff among all parameters like visual degradation, speed, encryption ratio, compression friendliness, format compliance and cryptographic security

Although an important and rich variety of video encryption algorithms have been proposed in literature, most of the algorithms are not secure against cryptanalysis attack.

Naïve algorithm provides highest level of security but it is very slow in nature and cannot be used in real time. Permutation based algorithms are generally faster but they do not provide sufficient level of security. Selective encryption algorithms reduce computational complexity by selecting only a minimal set of data to encrypt but their security and speed level generally vary based on which and how many parameters they encrypt.

Perceptual encryption algorithms are suitable for application like pay per view TV, video on demand where potential users like to see low quality video before buying them. So, these algorithms are not suitable for applications which demand high security.

It is difficult for a single algorithm to satisfy all performance parameters. So, selection of encryption algorithm always depends on requirements of application in use. By looking at table we can conclude that it is a challenge for researchers to design an encryption algorithm which maintains tradeoff among all parameters like visual degradation, speed, encryption ratio, compression friendliness, format compliance and cryptographic security.

The main idea we want visitors to our demo to take away is that it is possible to build privacy-preserving systems to share data on mobile devices without revealing any information until you are convinced the transaction is appropriate. We have demonstrated this by creating a system that enables video exchange using the video itself as a proof of presence.

No information is revealed until the participant chooses to release the video due to the application of a fully homomorphic cryptosystem. Additionally, the algorithms for determining similarity have been designed in such a way that they can run on a mobile phone in hundreds of milliseconds

**5.2. Future work**

We will bring several devices such as laptops and mobile phones that have our software preloaded on them that we will use to demonstrate the video sharing software.

In addition, we will also provide a QR code that will link to an APK that Android users will be able to install on their mobile phones so that they can also participate in the demo with their own devices and try it in other areas besides in front of our poster. We intend to connect to the venue’s Wi-Fi to perform the demonstration.

In this paper, we have proposed a methodology where by using the two algorithms AES and Blowfish [8] by integrating them and implemented an attempt to present a video encryption sharing.

Application for the Android based mobile devices. Although an important and rich variety of video encryption algorithms have been used, most of the algorithms defined are not completely secured.

This application allows users to send file or data in the video format to other android devices in a secure network. In this way, application can still access the data without reaching for user’s sensitive information. This application can be useful in many real time events and applications for an enhanced user support.

It is observed that the decryption process takes more time relatively as compared with the encryption process, which is acceptable because of the nature of the algorithms besides this the algorithms are implemented in limited resources.

The main advantage of this system is achieving the protection for data of video in android devices such as confidentiality for the secure end-to-end user communication and for simple user interface so that it is easy for users to interact.

he “Random Block Video Encryption” can be modified further using other encryption methods such as Hash key-based video encryption scheme for H.264\AVC, MPEG encryption algorithms, etc. A comparison and performance need to be tested for the best efficient algorithm implementation.

The project paper does not address the LIVE content streaming of videos. The scope could further be extended to the process of fetching LIVE video content securely by implementing an RBVE algorithm and performing user authentication, as this is an emerging factor within the field of video streaming.

**REFERENCES**

[1] Shiguo Lian, Dimitris Kanellopoulos, and Giancarlo Ruffo, “Recent Advances in Multimedia Information System Security,” International Journal of Computing and Informatics, Vol. 33, No.1, 2009, pp. 3-24

[2] Shiguo lian, Multimedia Content Encryption: Algorithms and Application, CRC Press, 2008.

[3] B. Furht, D. Socek, and A. M. Eskicioglu, “Fundamentals of Multimedia Encryption Techniques,” Multimedia Security Handbook, CRC Press, 2005

[4] ISO/IEC 13818:1996, Coding of Moving Pictures and Associated Audio (MPEG-2); Part 1: systems, Part2 : video.

[5] Mitchell J. L., Pennebaker W.B., fogg C.E. and LeGall D.J., MPEG Video Compression Standard, Chapman & Hall, 1996.

[6] ITU-T Rec. H.264/ISO/IEC 11496-10. Advanced Video Coding. Final Committee draft, Document JVT-E022, 2002

7] Iain E G Richardson. H.264/MPEG Part10, 2002. Available: <http://www.vcodex.com>.

[8] NIST: Data Encryption Standard, FIPS 46-3, 1999.

[9] NIST: Advanced Encryption Standard, FIPS 197, 2001.

[10] S. Jiang, Z. Li, P. Zhou, and M. Li, “Memento: An Emotiondriven Lifelogging System with Wearables,” ACM Trans. Sen. Netw., vol. 15, no. 1, pp. 8:1–8:23, Jan. 2019. [Online]. Available: <http://doi.acm.org/10.1145/3281630>

[11] <https://ieeexplore.ieee.org/abstract/document/9431029>

12] J. H. Cheon, A. Kim, M. Kim, and Y. Song, “Homomorphic Encryption for Arithmetic of Approximate Numbers,” Tech. Rep. 421, 2016. [Online]. Available: <https://eprint.iacr.org/2016/421>

[13] Microsoft SEAL (release 3.3), Jun. 2019. [Online]. Available: <https://github.com/Microsoft/SEAL>.

14] C. Gentry, “Fully homomorphic encryption using ideal lattices,” in Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC ’09. Bethesda, MD, USA: ACM Press, 2009, p. 169. [Online]. Available: http://portal.acm.org/citation.cfm? doid=1536414.1536440.

[15] K. Wu and B. Lagesse, “Do You See What I See? Detecting Hidden Streaming Cameras Through Similarity of Simultaneous Observation,” IEEE Pervasive Computing and Communications, p. 10, 2019.

[19] M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan, “Homomorphic Encryption Security Standard,”

[20] K. Yang, Z. Liu, X. Jia, and X. S. Shen, “Time-Domain AttributeBased Access Control for Cloud-Based Video Content Sharing: A Cryptographic Approach,” IEEE Transactions on Multimedia, vol. 18, no. 5, pp. 940–950, May 2016.

[21] B. Thuraisingham, G. Lavee, E. Bertino, J. Fan, and L. Khan, “Access control, confidentiality and privacy for video surveillance databases,” in Proceedings of the eleventh ACM symposium on Access control models and technologies - SACMAT ’06. Lake Tahoe, California, USA: ACM Press, 2006, p. 1. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1133058.1133061>

[22] J. Fan, H. Luo, M.-S. Hacid, and E. Bertino, “A Novel Approach for Privacy-preserving Video Sharing,” in Proceedings of the 14th ACM International Conference on Information and Knowledge Management, ser. CIKM ’05. New York, NY, USA: ACM, 2005, pp. 609–616, event-place: Bremen, Germany. [Online]. Available: <http://doi.acm.org/10.1145/1099554.1099711>.

[23] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar, “Efficient privacy preserving video surveillance,” in 2009 IEEE 12th International Conference on Computer Vision, Sep. 2009, pp. 1639– 1646.

[24] C. Dobbins and S. Fairclough, “A mobile lifelogging platform to measure anxiety and anger during real-life driving,” in 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Mar. 2017, pp. 327–332.

[25] C. Gurrin, A. F. Smeaton, and A. R. Doherty, “LifeLogging: Personal Big Data,” Foundations and Trends R in Information Retrieval, vol. 8, no. 1, pp. 1–125, 2014. [Online]. Available: http://dx.doi.org/10.1561/1500000033

[26] S. Jiang, Z. Li, P. Zhou, and M. Li, “Memento: An Emotiondriven Lifelogging System with Wearables,” ACM Trans. Sen. Netw., vol. 15, no. 1, pp. 8:1–8:23, Jan. 2019. [Online]. Available: <http://doi.acm.org/10.1145/3281630>

[27] S. Lian, J. Sun, and Z. Wang, “Perceptual Cryptography on SPIHT Compressed Images and Videos, “in Proceedings of IEEE International Conference on Multimedia and Expo,2004, Vol. 3, pp. 57-60.

[28] S. Lian, “Perceptual Cryptography on JPEG2000 Compressed Images or Videos, “ in Proceedings of International Conference on Computer and Information Technology, IEEE Consumer Society, 2004, pp. 78-83.

[29] C. Wang, H.-B Yu, and M. Zheng, “A DCT based MPEG-2 Transparent Scrambling Algorithm”, IEEE Transactions on Consumer Electronics, Vol. 49, No. 4, 2003,pp. 1208-1213.

[30] W. Zeng and S. Lei, “Efficient Frequency Domain Selective Scrambling of Digital Video,” IEEE Transactions on Multimedia, Vol. 5, No. 1, 2003, pp. 118–129.

[31] C. Bergeron and C. Lamy-Bergot, “Compliant Selective Encryption for H.264/AVC Video Streams,” in Proceedings of the 7th IEEE Workshop on Multimedia Signal Processing ,2005, pp. 1–4.

[32] Shiguo Lian, Zhongxuan Liu, Zhen Ren and Haila Wang, “Secure Advanced Video Coding Based on Selective Encryption Algorithms,” IEEE Transaction on Consumer Electronics, Vol. 52, No. 2 ,2006, pp. 621-629.

[33] Shunjun Li, Guanrong Chen, Albert Cheung, Bharat Bhargava, and Kwok-Tung Lo, “On the Design of Perceptual MPEG Video Encryption Algorithm”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, No. 2, 2007, pp. 214-223.

[34] M. Pazarci and V. Dipcin, “A MPEG2-Transparent Scrambling Technique”, IEEE Transactions on Consumer Electronics, Vol. 48, No. 2, 2002, pp. 345-355.

[35] S. Lian, X. Wang, J. Sun, and Z. Wang, “Perceptual Cryptography on Wavelet Transform Encoded Videos, “in Proceedings of IEEE International. Symposium on Intelligent Multimedia, Video and Speech Processing, 2004, pp. 57-60

[36] C. Shi and B. Bhargava, “A Fast MPEG Video Encryption Algorithm,” in Proceedings of the 6th ACM International Conference on Multimedia, 1998, pp. 81–88.

[37] C. Shi and B. Bhargava, “An efficient MPEG video encryption algorithm,” in Proceedings of the 17th IEEE Symposium on Reliable Distributed Systems 1998, pp. 381–386.

[38] C. Shi, S. Y.Wang, and B. Bhargava, “MPEG Video Encryption in Real-time using Secret Key Cryptography,” in Proceedings of the International Conference on Parallel and Distributed Processing Algorithms and Applications, 1999,pp. 191–201.

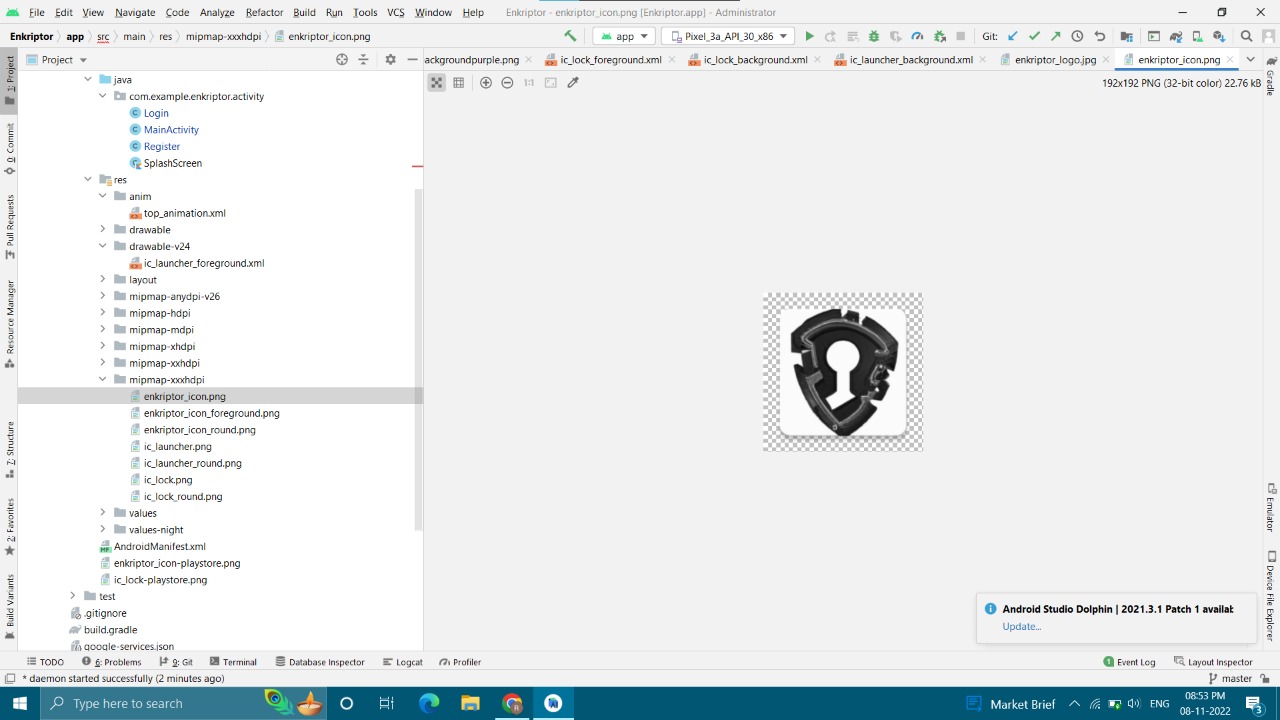
[39] C.-P. Wu and C.-C. J. Kuo, “Fast Encryption Methods for Audiovisual Data Confidentiality,” in Proceedings of SPIE, 2001, Vol. 4209, pp. 284–295.

[40] C.-P. Wu and C.-C. J. Kuo, “Efficient Multimedia Encryption via Entropy Codec Design,” in Security and Watermarking of Multimedia Contents ,in Proceedings of SPIE,2001, Vol. 4314, pp. 128–138. [25] D. W. Gillman and R. L. Rivest, “On breaking a Huffman code,” IEEE Transactions on Information Theory, vol. 42, no. 3, 1996,pp. 972–976.

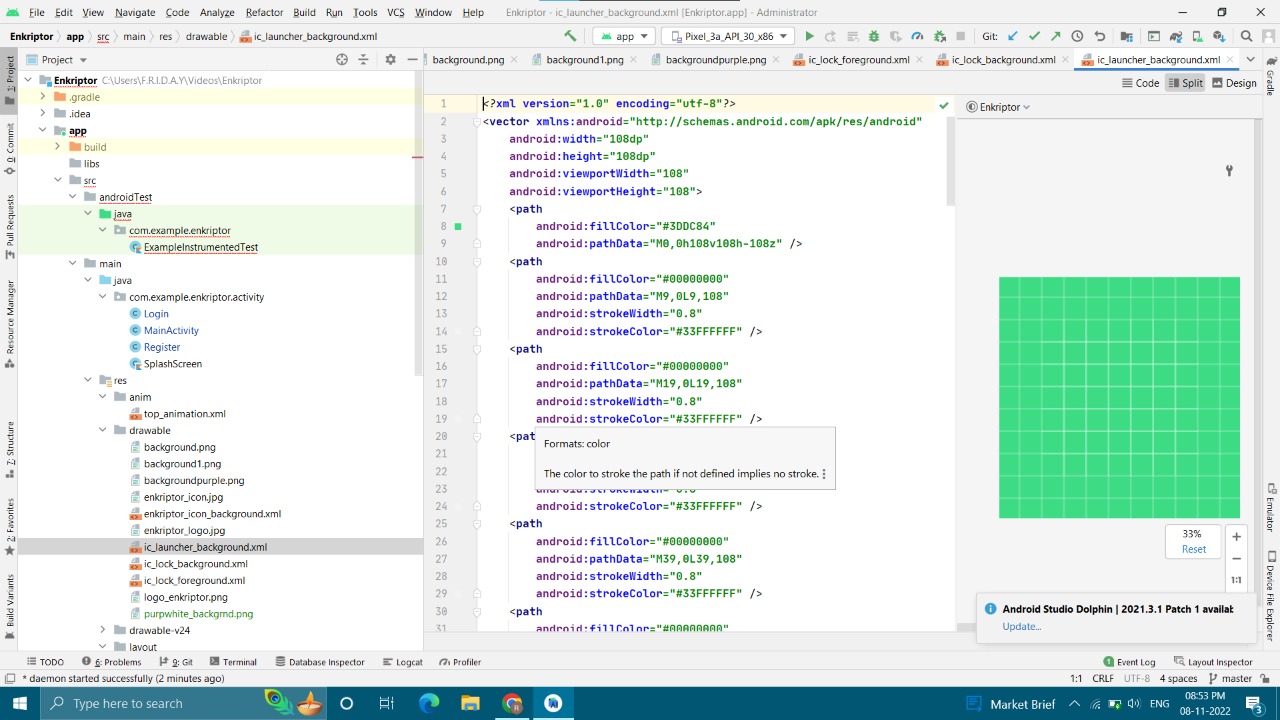
[41] C. Shi and B. Bhargava, “Light Weight MPEG video Encryption Algorithm”, in Proceeding of the International Conference on Multimedia, 1998,pp. 55-61.

[42] Hao Wang and Chong-wei Xu, “A New Lightweight and Scalable Encryption Algorithm for Streaming Video over.

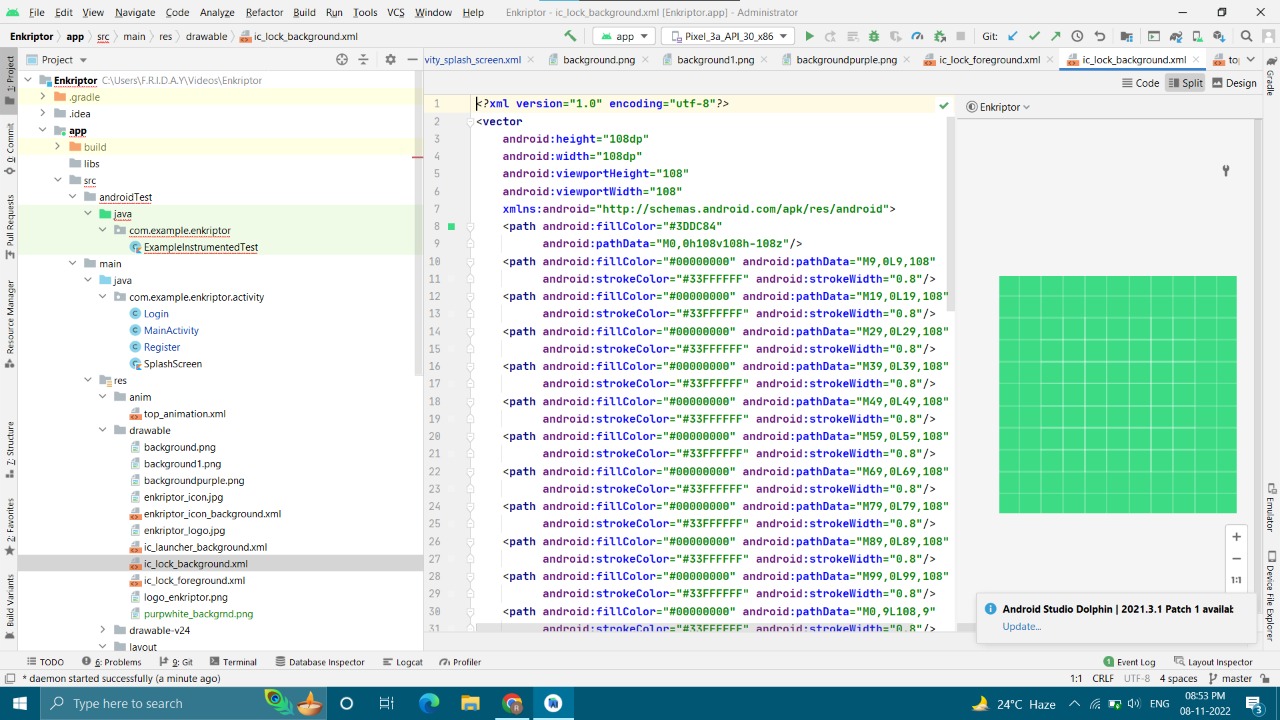
**APPENDIX**



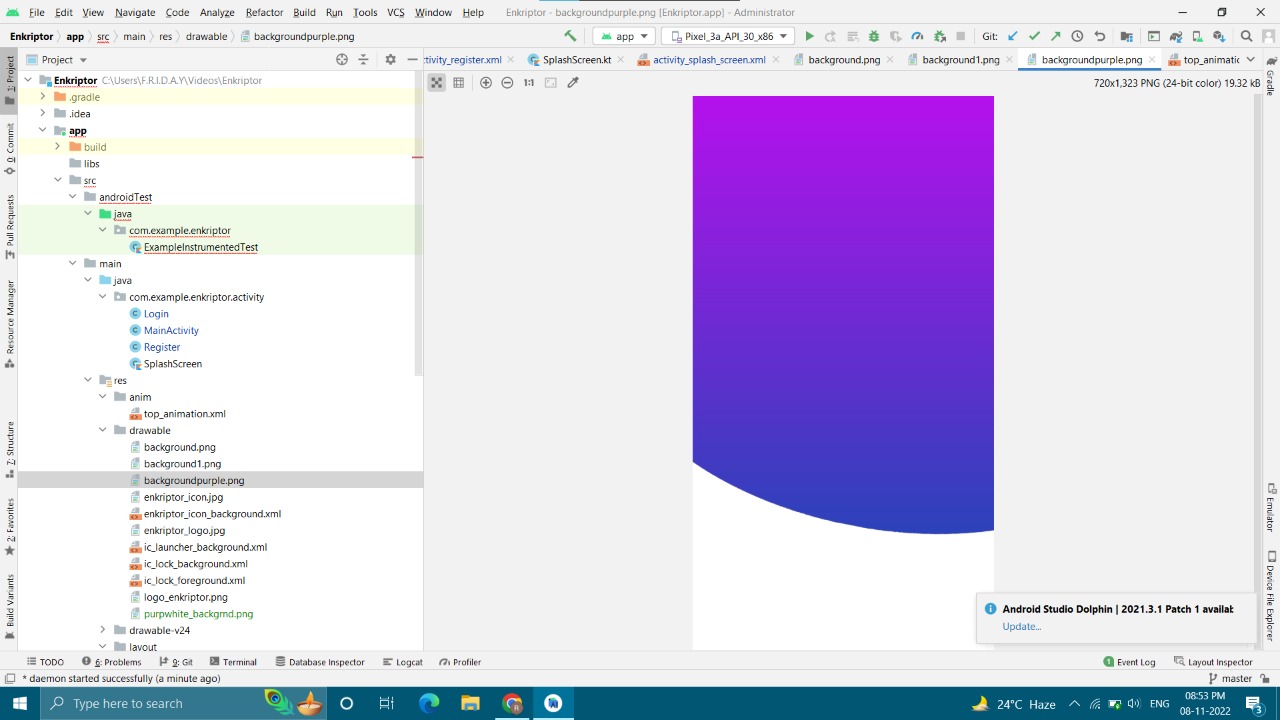
Enkriptor logo Icon PNG File



Ic\_background.xml



IC\_lock Backround.xml



Background of Register and Main Activity

## 

Enkriptor

## 

## Colors.xml

## 

## String.xml

## 

## Nightthemes.xml

## 

## Manifest.xml

## 

## ExampleUnitTest.java file

## 

## Gradlesw

## 

## Gradlesw

## 

## Gradlew.bat

## 

## Gradlesw.bat

## 

## Local.Properties

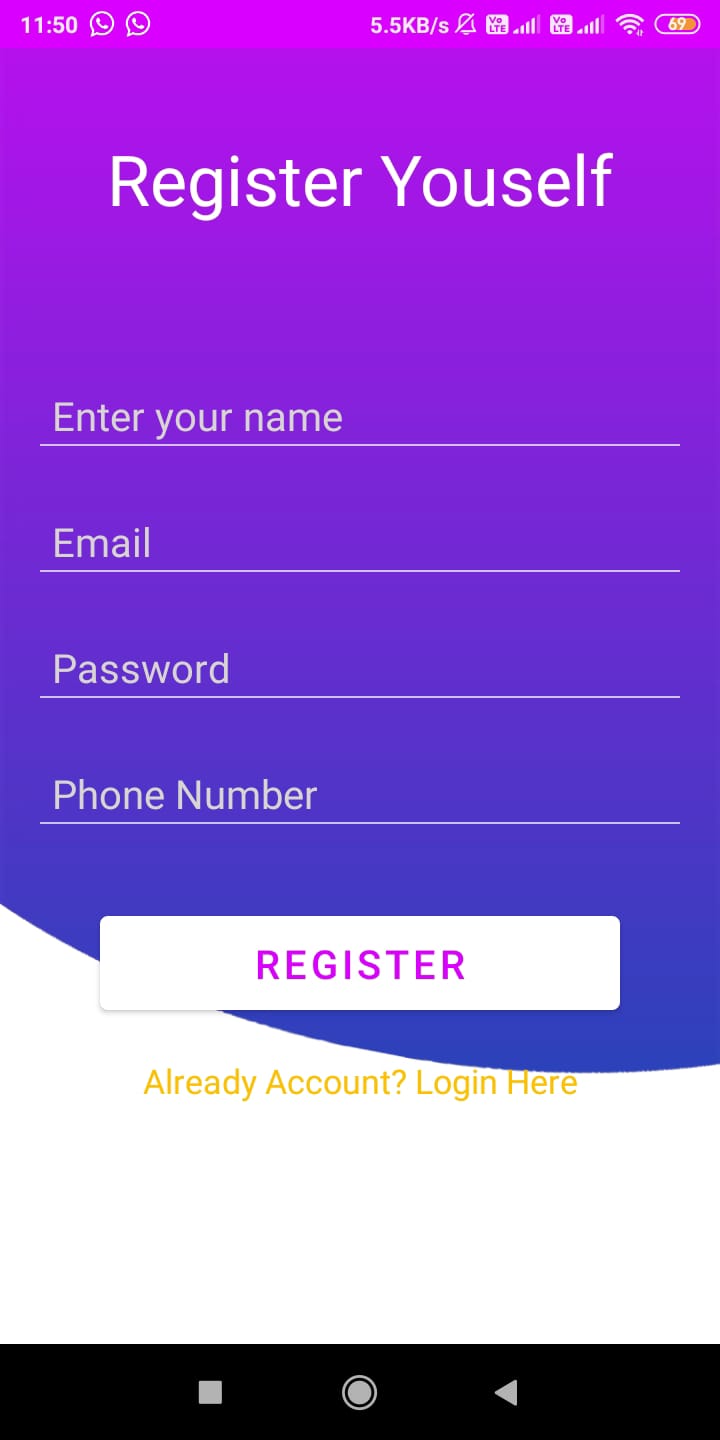
## Git repo-action

**USER MANUAL**

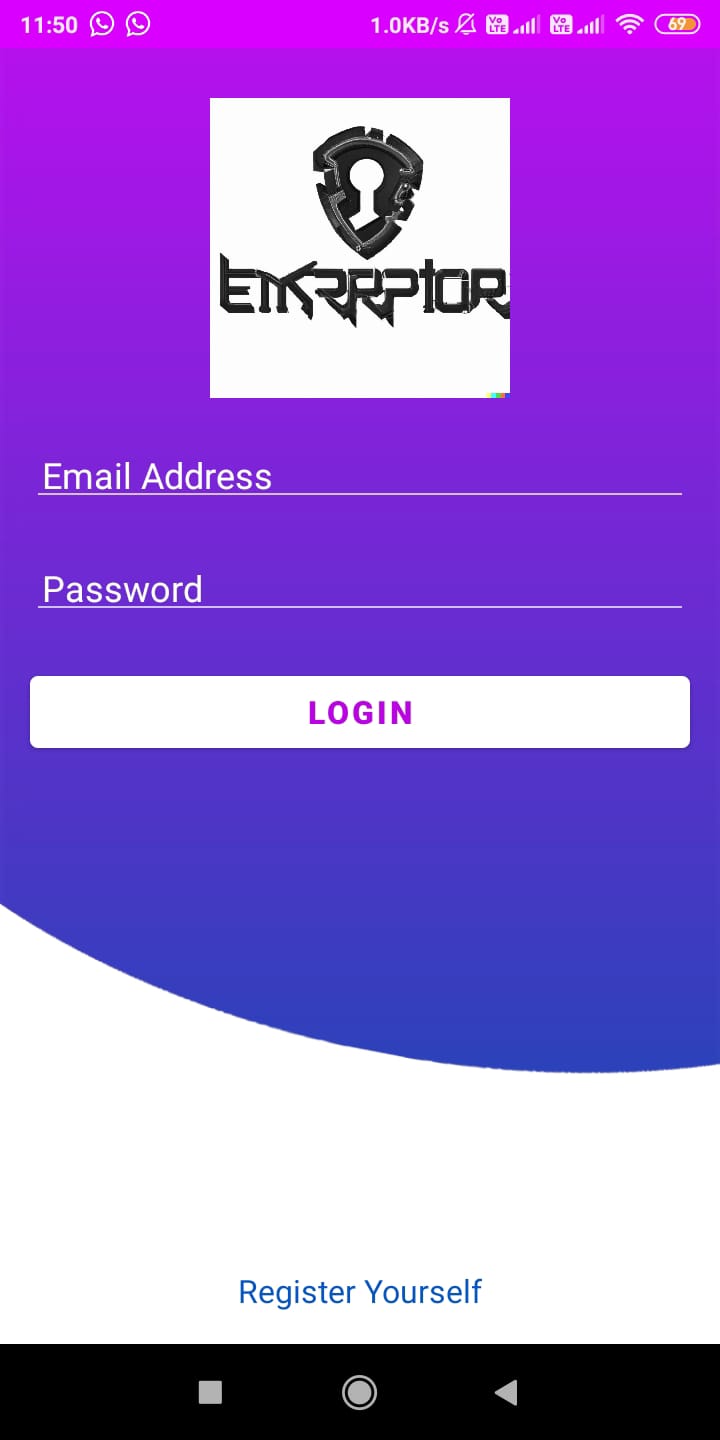
SPLASH SCREEN:

****

**Register Activity:**

****

**Login Activity:**

****