SECURITY ENHANCEMENT IN EXISTING ANDROID PERMISSION BASE MODEL

A Dissertation II

Submitted

By

Pankaj Laheri

To

Department of Computer Science and Engineering

In partial fulfillment of the Requirement for the

Award of the Degree of

Master of Technology in Computer Science and Engineering

Under the guidance of

Ms. Anu Garg

(Assistant Professor, Lovely Professional University)

(May 2015)

PAC Approval Page

INDIA'S LARGEST UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education, Transforming India

School of: LFTS – (CSE-ECE)

DISSERTATION TOPIC APPROVAL PERFORMA

Name of the Student: PANKAJ LAHERI Registration No: 11306892

Batch: 2013-2015 Roll No. RK2308 A11

Session: 2014 Parent Section: K2308

Details of Supervisor: Designation: ASSISTANT PROFESSOR

Name ANU GARG Qualification: M.TECH

U.ID 16829 Research Experience: 2 years

SPECIALIZATION AREA: NETWORKING & SECURITY (pick from list of provided specialization areas by DAA)

Applications

PROPOSED TOPICS

1. Enchancing the permission-based systems architecture for mobile device.
2. Enforcing multiple security policies for Android Virtualization.
3. Detecting user usage attack based on system model ith learning engine.

Signature of Supervisor

PAC Remarks: Topic 1 is approved.

APPROVAL OF PAC CHAIRPERSON: Signature: Date: 19/9/17

*Supervisor should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)

*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.

*One copy to be submitted to Supervisor.

ABSTRACT

Android operating system has shared around 84 % of smartphone market for the second quarter of 2014 with 300 million smartphones running Android, still growing better than other mobile operating system. As due to open source large amount of applications are developed and downloaded on regular basis from android play store and from other sources. It is estimated that around one million application downloads are made on every single day from various sources. As increase of applications betters the user's experience but it also increases the user's files security issues. As large amount of malware applications are made to steal user's personal information. Therefore a proposed system is needed to secure these personal data. As cryptography techniques are one of the widely used methods to secure our data. Our proposed system also capable to detect if any process is granting access to these personal files.

DECLARATION

I hereby declare that the dissertation II entitled "**Security Enhancement in Existing Android Permission Model"** submitted for the M.Tech degree is entirely my original work and all the ideas and references have been dully acknowledged. It does not contain any work for the award of any other degree or diploma.

Date: _____

Pankaj Laheri

Reg. No. 11306892

CERTIFICATE

This is to certify that **Pankaj Laheri** has completed M.Tech dissertation-II titled **"Security Enhancement in Existing Android Permission Model"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma.

The dissertation proposal is fit for the submission and the partial fulfilment of the conditions for the award of **M.Tech in Computer Science & Engineering.**

Date:

Signature of Advisor
Name: Ms. Anu Garg
UID:

ACKNOWLEDGEMENT

The report has been written with the kind guidance and support of my mentor. The satisfaction and happiness that accompany the successful completion of any task would be incomplete without mentioning the names of people who made it possible, whose constant guidance and encouragement crowns all efforts with our success.

I would like to present my deepest gratitude to **Ms. Anu Garg** for her guidance, advice, understanding and supervision throughout the development of this dissertation study. I would like to thank to the **Project Approval Committee members** for their valuable comments and discussions. I would also like to thank to **Lovely Professional University** for the support on academic studies and letting me involve in this study.

Contents

LIST OF FIGURES

Chapter 1

INTRODUCTION

In this chapter, we are discussing about Android operating system, its various versions, working of framework. After this we are listing its features with its security model and its inbuilt permission base model.

1.1 Introduction to Android Operating System

Android is an operating system designed for mobile devices such as smartphones and tablets. It is developed by Google under Open Handset Alliance (OHA) created on 5, Nov, 2007.The Open Handset Alliance is basically a business alliance with purpose to develop open mobile standards. Open Handset Alliance has nearly 80 members having mobile operators, manufacturers, semiconductors firms and commercialization companies [1]. As all members shares a commitment to expand the open platform development. This results as companies can use the Android operating system and use it in their mobile devices.

Android is a Linux based operating system having java as a programming interface. Middleware, APIs, libraries are written in C language. Android uses the Dalvik Virtual Machine having just-in-time compilation to run the compiled Java code.

Android accompanies an Android market which is an online programming store. It was created by Google. It permits Android clients to choose, and download applications grew by outsider engineers and utilization them. There are around 2.0 lakh, applications and gadgets accessible available for clients.

Android applications are composed in java programming dialect. Android is accessible as open hotspot for engineers to create applications which can be further utilized for offering as a part of android business. There are around 200000 applications produced for android with more than 3 billion and more downloads. Android depends on Linux variant 2.6 for core framework administrations, for example, security, memory administration, process administration, system stack, and driver model. For programming advancement, Android gives Android SDK.

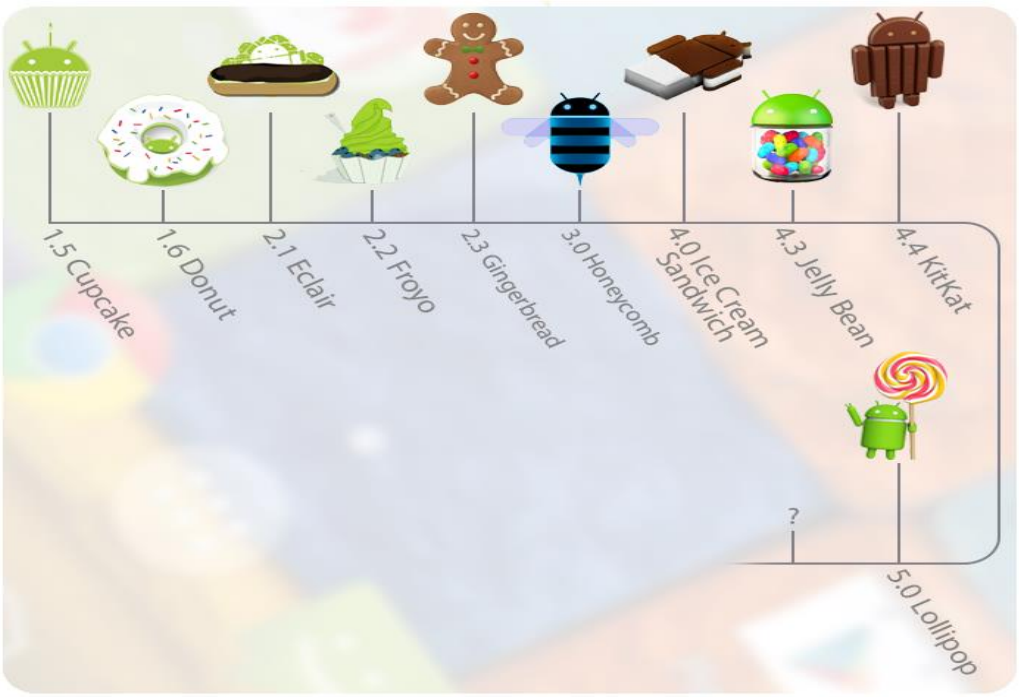

Figure 1: Android Operating System Versions [2]

1.2 Android Framework

Android operating system is basically organized in a stack of layers, as each layer has different functionality and provides services to the layer sitting to its layer above it. The heart of Android Operating system is Linux kernel 2.6 (now it's running on 3.4 or higher) which is modified by Google and add some basic architectural features. Above this kernel layer native libraries and Android runtime is placed. As these are modules of code and are compiled down to native machine code. These run directly on processor and outside Dalvik Virtual Machine.

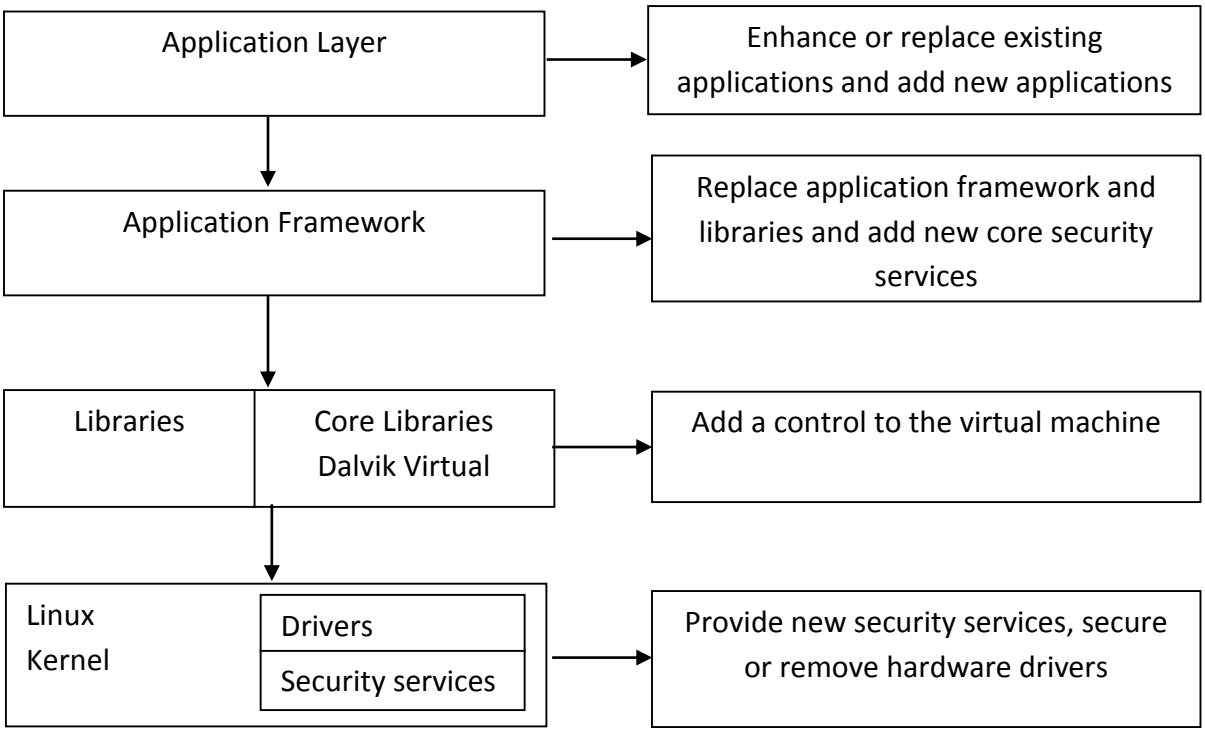

Figure 2: Android Security Framework

Native libraries run as processes under the Linux kernel. Android Runtime consist of Dalvik virtual machine and core java libraries. As Dalvik virtual machine is optimize for mobile as it run fast in low resource environments. Every individual application runs in its own instance of Android Runtime, having instance of Dalvik VM and core libraries.

1.3 Android Features

- Android code under the Apache License, a free software and open source license.
- It relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.
- A very important feature of Android OS is, it is open source nature, develop new application or update existing application.
- Each Android app runs within its own virtual machine and each virtual machine is isolated in its own Linux process.
- Each app is given unique user and group IDs
- All applications have full access to phone capabilities.

- All applications are permissions- based.
- It allows access to core mobile device functionality through standard API calls.
- E.g.: applications can call core functionality such as making calls, sending text messages, using camera.
- A powerful SDK is available for development that contains libraries, tutorials, sample code and emulator.
- Should have no costs for using the platform, develop applications for the platform or publish own developed applications.

1.4 Market offer

As a designer, have a chance to create applications for a genuinely new market that is blasting regularly. With such a variety of clients, its never been less demanding to compose an application that can be downloaded and utilized by genuine individuals. Android gives engineers an approach to create extraordinary, imaginative applications and get those applications in the hands of user's. Clients don't need to go seeking the Internet to discover an application to introduce. They just basically go to the Android Market that is preinstalled on their gadget, and they have entry to all applications. Since the Android Market comes preinstalled on most Android gadgets, clients normally hunt the Android Market down the majority of their application needs.

1.5 Time to market

With all the application programming interfaces (APIs) that Android comes pressed with, its anything but difficult to grow full-included applications in a generally brief timeline. After joined with the Android Market, transfer the applications and distribute them. Dissimilar to other versatile commercial areas, the Android Market has no application endorsement process. In the Android Market, there are no such confinements its dependent upon clients to assess the applications they purchase upon establishment. In fact, it is simple for the client to create and distribute the applications inside Google's terms of administration.

1.6 Open stage

The Android working framework is open stage, implying that it is not attached to one equipment producer or one supplier. The openness of Android is permitting it to pick up piece of the overall industry rapidly. The code fundamental the open stage is accessible for investigation by clients and engineers. The open-source code permits telephone makers to make custom client interfaces and include fabricated in highlights to a few gadgets. It's workable for everybody to get to the crude Android source code.

1.7 Cross- similarity

Android can run on numerous gadgets with diverse screen sizes and resolutions. Other than being cross-good, android accompanies the apparatuses that assistance to create cross-perfect applications. Google permits the applications to run just on perfect gadgets. On the off chance that application obliges a front-confronting cam, for instance, just telephones with a front-confronting cam will have the capacity to see application in the Android Market. Android gadgets need to be affirmed good to have admittance to the Android Market. Similarity guarantees that the applications can run on all gadgets.

In the wake of sending the Android, in this area we will depict the administrations of working framework. The Open Handset Alliance (OHA) is a confederation of 50 Telecoms, portable equipment, and programming organizations. Google was among the first in the portable OS group to open versatile OS's by adding to the Android Platform, supporting measures and distributed APIs. The working framework is massively engaging both handset fabricates and programming engineers in light of the fact that they can make their own form of Android while last can get their applications onto the android market. Great Technology utilizes Android gadgets to join with their corporate venture so representatives can get to organization assets by means of a safe holder in the customer which differentiates shielded undertaking information from individual information and applications put away on the cell phone. It incorporates contributed programming and other licensed innovation from its part organizations and makes it accessible to engineers through the open source group.

For instance, when faulty charges are connected against a client's ledger, the budgetary establishment quickly makes an impression on the client's telephone asking for area and buy data. The client has the chance to take after responsive convention and sanction the exchange.

1.8 Linux security model

As much of Android security relies on Linux kernel. Linux system security central to users and groups. To every user in Linux operating system a unique user ID (UID) is assigned. These users added to group and to each group again a unique ID is assigned (GID). In Linux system each resource has an owner is identified by UID. Owner can alter permission on it and has primary responsibility on it. Each Linux resource has three sets of permissions a) owner- that applies only to its owner. b) group- applies to its group users. c) world- apply to anyone. Each permission has these rights a) Read b) Write c) Execute [4]. The Least Privilege Principle of Linux states that entities have only minimum access to resources. Thus entities have read permission does not have write permission or vice versa. If entity does not have any permission means simply it does not have.

On install time Linux creates a unique UID and it assigns to the newly installed apps. The app runs under that particular UID and become owner of all app resources. Linux prevents for accessing the process or memory of other apps. Therefore provides separation between apps. At install time of app android makes a directory under path /data/data/App_package_name. Android sets the UID as owner of directory. Set of permissions created and assigned to this folder. Inside this directory Android creates /file, /database directories. As each file created by apps set as private and not declared public unless developer do so.

1.9 Android permissions

Android isolates its apps so that one app cannot use other apps resources. But sometimes it is need to share resource among apps. To do so Android creates a permission base request model. App can access other apps resources and services if it is granted the permission set too. The permission request model is "all or nothing" process [5]. Either to accept all permission then only application will install otherwise it is not installed. Android permission request model has main goal: a) it informs user about all sensitive and dangerous things that app can do before installation. User has to read all permissions and understand its effect. As developers has to set

or grant only minimum number of app permissions not more than it needed. User and developer has important role so that the effect of permission request model become effective.

1.10 Protecting app stored data

Some secure approach would be like minimize the use of those applications which accesses sensitive and personal data. Not storing the personal data on SD card, as because no special protection is given to this storage. It even bypass Linux base isolation function. Threat occurs when user has root access to system. This will result unrestricted and complete access to device memory, storage while device in running. And isolation functionality does not exist when applications run with root identity.

1.11 Using Cryptography

Android provides a wide area to provide security using cryptography. Cryptographic algorithms use either symmetric or asymmetric encryption. Encryption provides confidentiality for sensitive information by using one key for encrypting and decrypting the data. As symmetric encryption algorithms performs fast and provides good level of protection. In these encryption algorithms the concern about the keys, if the key get lost the encrypted data cannot be recovered. As NIST recommends AES (Advanced Encryption Standard).
The strength of these based upon length of key. Longer the key more protection is offered. Asymmetric keys encryption offers two different keys one for encryption and one for decryption. Strength of this encryption also depends upon length of keys. RSA provides keys length at least 2048 bits long to provide protection as compared to AES-256.

Asymmetric algorithms are generally slow than that of symmetric algorithms, therefore data should be minimum for encryption and decryption. Hash function takes message of any size and gives fixed size digest as hash (random sequence of bits).

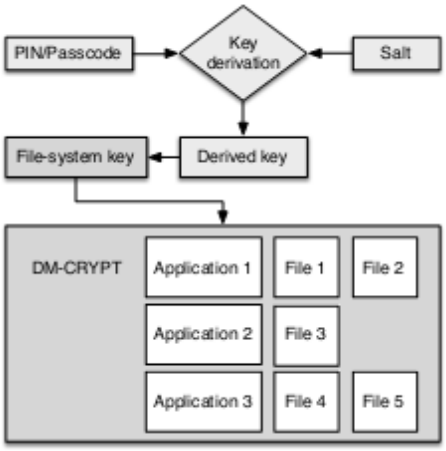

Figure 3: File System encryption.

SHA-256 produces 256 digest. Hashes functions are one way functions go only from message to digest not vice versa. Hash functions are used to validate credentials. As user's credentials is very sensitive, so hash functions generates its digest. User is asked to fill password hash of the newly typed password is compared with store digest.

If it match user is authenticated. As another way is encrypting the user’s password and on authentication the stored value which is decrypted first is checked with key provided by user. If it matches user authenticated. Hash functions stores and validates user's credentials. Therefore if unauthorized persons access the stored credentials somehow, he cannot get password out of it.

Chapter 2

REVIEW OF LITERATURE

In this chapter, we are reviewing various published literatures which are basically worked on optimizing android security.

Raluca-Alina et al [5] 2014. Privacy Application is implemented in this paper. This application protects the user's private data, SMS, Call logs. Through this application an interface is provided to authenticate user by username and password each it uses this application. Hash function (SHA-256) is applied on user's credentials resulting in a digest which is then stored in database. Each time after authentication digest is generated and checked with the stored digest for authentication. Password based encryption is used in which encryption key is derived from user's password using. Advance Encryption Standard (AES) technique. Key derivation function is used for deriving the key value. User's password is needed whenever encryption or decryption of data is needed.

Stephan Heuser et al [6] OCT-2014. ASM provides programmable interface for defining new reference monitors. ASM framework allows the users to use this implementation without making any changes in their firmware. Provides sufficient protection to customers, enterprise and government. ASM have reference monitors that are implemented by ASM apps. Whenever a protection operation occurs, android first contacts its security module (ASM app) that whether that operation should proceed or not so that the personal information of user should not use, shared anonymously by any third party application. As these reference monitors are placed within ASM Bridge which is further connected to ASM LSM for kernel level authorization. In this paper ASM performance is evaluated on two applications a) MockDroid b) AppLock. ASM without ASM apps have overhead of 3.34% whereas ASM with ASM apps have overhead of 9.33%.

Muneer Ahmad Dar et al [7] APRIL-2014. This paper proposed an enhanced security framework which enhances the security of Android file system by restricting the apps whose behavior matches with the malware. It secures the user's data using cryptographic algorithms.

Advance Encryption Standard (AES) technique is applied to all personal files stored in the device so that to prevent modification and access of data from any unauthorized app. If an app wants to access those files it first checked through various Malware detection techniques: a) Signature Matching. b) Heuristics c) Hashes. If security API finds any vulnerability in app it restricts the access mechanism of app and hence denies its access to the file. If it fails to find any vulnerability in app, then the granted file is decrypted first and allow app to access that file.

Divya sukhija et al [8] 2014. Gave a review on certain cryptographic algorithms author has discussed about two techniques AES (Advanced Encryption standards) and DES (Data Encryption Standards) where in author reviewed these algorithms in details. Author has given important facts about symmetric as well as asymmetric algorithm and has briefly shown how and why we are using such kind of algorithm. Author has told about brief the history of DES and AES and has also discussed Strengths and Weakness for the same.

Chanhee Lee et al [9] 2013. Goals to achieve and proposed mechanisms. a) Preventing root level attack by privilege applications. (Root level attack). Proposes RPP (Root privilege protection): Keeps track of list of trusted application having root level privilege. It detects and responds to malware which tries to acquire root level privilege. b) Protecting system resources from illegal usage by malicious application. Proposes RMP (Resource misuse protection): Track list of critical resources and protect them for illegal exploitation. c) Protecting user's privacy or personal information. Proposes PDP (Private data protection): Keeps personal information safe by enforcing strict access control mechanism (least privilege principle) so that even privilege application cannot access user's private data. It introduced two data structure as: White List having listed of trusted programs CriticalList have list of critical resources. Conclusion: This framework is effective to prevent root attacks and misuse of user/system resources.

Hammad Banuri et al [10] 2012. This framework monitors dynamic behavior of application during runtime by using Security enhanced android framework (SEAF). It monitors the behavior of permission patterns during execution of particular application. As there is limitations in android- a) Android permission model lacks customization of permission and

follows all or nothing policy. As user has to accept all permissions for installation or deny. b) Android does not investigate runtime behavior of application. 1. Policy evaluator: responsible for runtime evaluation of application. It checks the permission request file forwarded by package manager service to evaluate mode of application (Restricted or Unrestricted). Methods used: check mode (), checkpermissionstatus (), evaluatepolicy(). 2. Permission manager: It provides GUI for user to customize application permission. And allow user to assign application mode either restricted or unrestricted. Conclusion: This framework not only overcome the limitations of android but also built some realistic policies which stores and monitor the pattern access by an application during runtime execution.

Zhaohui Wang et al [11] 2012. Mobile devices stores Personal Identifiable information (PII) for application. These applications uses data related to SMS, emails, location based information etc. Therefore this paper proposes FUSE (File system in user space) to protect data on devices. It is a library which implements file system in user space and acts as bridge to kernel interface. It uses ENcFS which provides file system encryption. ENcFS runs over base file system e.g. (ext4, ext3). Openssl is used to provide cryptographic functions and which is integrated in ENcFS. ENcFS uses AES encryption technique and uses 128-256 bit key length value. Key length increases to 128-256 bit, the overhead loss increases from 10% to 15 %. As by optimizing the block size and I/O mode performance overhead increases from 20% to 57%.

Rohan Rayarikar et al [12] Encryption is one of the major important process to make our data confidential. In this paper author has created a platform from where message is encrypted before it is transmitted over the network. Various encryption algorithm are in the market to secure our private data from any third party users e.g. AES, DES, RCA. As AES algorithm is widely accepted algorithm and it has very secure encryption pattern. Therefore message encrypted by this algorithm is resilient to Brute-Force and pattern attacks. This application meets the speed and compactness as application size is only 50 kb and user experiences no delay in using this program. This application also detects if any message has been corrupted or has been tempered during transmission.

Adam Skillen et al [13] Data confidentiality is very effective in preserving the encryption data. But in some situation user may be forced to disclose decryption keys. Therefore paper proposes

a system called Mobiflage to address obstacles that can compromise plausibly deniable encryption (PDE) in a mobile environment. This system hides the encrypted volumes within random data in devices free storage space and design new countermeasures for specific threats to mobile systems. They provide two implementations for the Android OS, to assess the feasibility and performance of Mobiflage on different hardware profiles. MF-SD is designed for use on devices with FAT32 removable SD cards. With Mobiflage, we explore design and implementation challenges of PDE for mobile devices, which may be more useful to regular users and human rights activists.

Yajin Zhou et al [14] This paper, focus on the Android platform and aim to systematize or characterize existing Android malware. Particularly, with more than one year effort, they had managed to collect more than 1,200 malware samples that cover the majority of existing Android malware families. They are characterize them to various aspects, including their installation methods, activation mechanisms as well as the nature of carried malicious payloads. Based on the evaluation with four representative mobile security software, their experiments show that the best case detects 79.6% of them while the worst case detects only 20.2% in their dataset. These results clearly call for the need to better develop next-generation anti-mobile-malware solutions.

Chapter 3

PRESENT WORK

In this chapter, we are discussing our scope of study, what we are included and what we are taken out of our scope. Including its enhancement helps in increasing android security.

The Scope of my study covers tackling security issues regarding storage of user's files and about its secure access to only authorized or valid user and application. Scope of our study covers encryption and decryption on user files by using encryption technique and validating user credentials as username and password using hash function. Therefore from this approach only the authorized user have access to the encrypted files. The credentials as username and password stored in the form of digest in the device database. Android Full-disk encryption will be part of our scope of study. As it will degrade the overall performance of operating system. Our Proposed work scope is limited to SD-card storage encryption and decryption only by working on detection if any process access the private directory or files of application.

Proposed system provides enhancement in security model of android in:

a) Mobile security

b) File system Security

c) Android Framework Security.

3.1 Problem Formulation:

Android operating system has grown rapidly from past few years and version updates comes in every two and a half month. As it is still developing, android operating system provides encryption of files on whole File disk once which makes whole system performance slow. Therefore it is concluded from many previous research papers that we can provide security to an individual module of File system which does not make performance degradation too.

The main aim of entire research work is to provide more secure file system for mobile devices and protecting the user's data from unwanted access from any third party application. As large amount of malware applications increasing day by day and user is unaware of the behavior of application running in backend of their device as these type of application's aim is to steal user's personal information and use the information against them. Therefore a proposed system is needed to secure these personal data and detection of those process which are granting access to data or files. Hence, enhancing security in Android File system.

3.2 Objective of Study

Key objectives:

a) To study the various attacks on file in android storage space.

b) To apply encryption techniques on the user's confidential files and validate the user using hash function.

c) To restrict the unauthorized access of any third party application for User's personal data.

d) The proposed system is not limited up to this functionality as it will detect any process which trying to access the private directory or its files anonymously.

e) To enhance security in Android File system without making Full-disk encryption to whole of Android storage space.

3.3 Research Methodology

In this chapter, we are discussing our research methodology including our proposed model and flow chart with every steps. And algorithm hash values as well as summary of tools used in implementing research work.

Data or Files not stored in encrypted form in mobile storage. Therefore third party or malware applications modify and access this information easily. So there is a need to secure the data using cryptography and also there is a need to detect and block those applications.

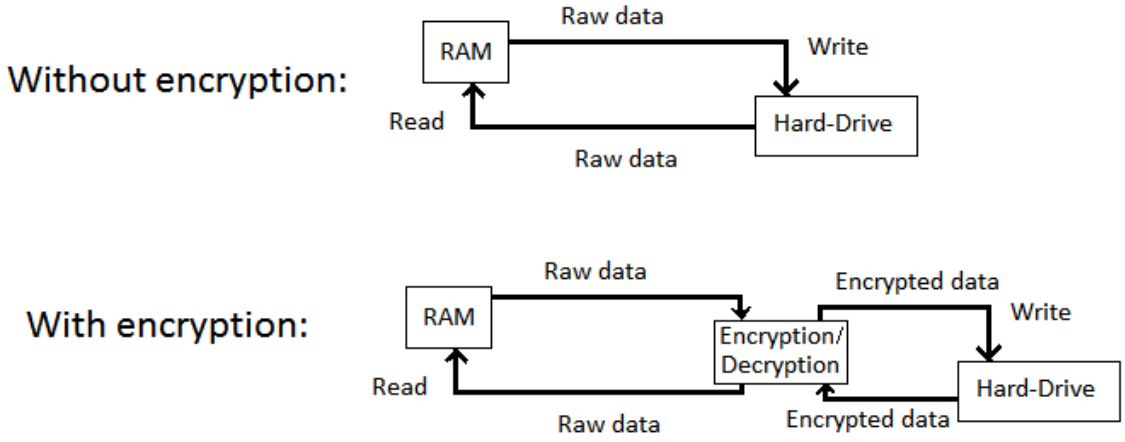

Figure 4: Files with and without Encryption. [15]

3.4 Cryptographic technique

Cryptographic algorithms use either symmetric or asymmetric encryption. Encryption provides confidentiality for sensitive information by using one key for encrypting and decrypting the data. As symmetric encryption algorithms performs fast and provides good level of protection. In these encryption algorithms the concern about the keys, if the key get lost the encrypted data cannot be recovered. As NIST recommends AES (Advanced Encryption Standard). The strength of these based upon length of key. Longer the key more protection is offered. Asymmetric keys encryption offers two different keys one for encryption and one for decryption. Strength of this encryption also depends upon length of keys.

a) AES:

AES embodies three piece figures, AES-128, AES-192 and AES-256. Every figure scrambles and unscrambles information in pieces of 128 bits utilizing cryptographic keys of 128-, 192- and 256-bits, individually. (Rijndael was intended to handle extra piece sizes and key lengths, yet the usefulness was not received in AES.) Symmetric or mystery key figures utilize the same key for scrambling and decoding, so both the sender and the recipient must know and utilize the same mystery key. Every key length are considered sufficient to secure characterized data

up to the "Mystery" level with "Top Secret" data obliging either 192- or 256-bit key lengths. There are 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys - a round comprises of a few handling steps that incorporate substitution, transposition and blending of the info plaintext and change it into the last yield of ciphertext.

AES is taking into account an outline guideline known as a substitution-stage system, blend of both substitution and change, and is quick in both programming and hardware. Unlike its ancestor DES, AES does not utilize a Feistel system. AES is a variation of Rijndael which has an altered square size of 128 bits, and a key size of 128, 192, or 256 bits. By differentiation, the Rijndael detail as such is indicated with piece and key sizes that may be any of 32 bits, both with at least 128 and a most extreme of 256 bits.

AES works on a 4×4 section significant request framework of bytes, termed the state, albeit a few renditions of Rijndael have a bigger square size and have extra sections in the state. Most AES counts are done in a unique limited field.

The key size utilized for an AES figure indicates the quantity of redundancies of change adjusts that change over the data, called the plaintext, into the last yield, called the ciphertext. The quantity of cycles of redundancy are as per the following:

a) 10 cycles of reiteration for 128-bit keys.

b) 12 cycles of reiteration for 192-bit keys.

c) 14 cycles of reiteration for 256-bit keys.

Each round comprises of a few preparing steps, every containing four comparative yet distinctive stages, including one that relies on upon the encryption key itself. An arrangement of opposite rounds are connected to change ciphertext again into the first plaintext utilizing the same encryption key

1) KeyExpansions

Round keys are gotten from the figure key utilizing Rijndael's key calendar. AES obliges a different 128-bit round key piece for each round in addition to one more.

2) InitialRound

a) AddRoundKey: every byte of the state is joined with a piece of the round key utilizing bitwise xor.

3) Rounds

a) SubByte: a non-direct substitution step where every byte is supplanted with another as per a lookup table.

b) ShiftRows: a transposition step where the last three lines of the state are moved consistently a specific number of steps.

c) MixColumns: a blending operation which works on the sections of the state, joining the four bytes in every segment.

d) AddRoundKey

4) Last Round (no MixColumns)

a) SubBytes

b) ShiftRows

3.5 Hashing Functions

Hash function takes message of any size and gives fixed size digest as hash (random sequence of bits) SHA-1 produces 160 bit digest. Hash functions are one way functions; go only from message to digest not vice versa. Hash functions are used to validate credentials. As user's credentials is very sensitive, so hash functions generates its digest. User is asked to fill password hash of the newly typed password is compared with store digest. If it match user is authenticated. As another way is encrypting the user's password and on authentication the stored value which is decrypted first is checked with key provided by user. If it matches user authenticated. Hash functions stores and validates user's credentials. Therefore if unauthorized persons access the stored credentials somehow, he cannot get password.

3.6 Tools:

Various tools required in implementation of the application such as Android SDK is used to create the application and to store the application database SQLite is use.

3.6.1 Android SDK:

Software development kit which is use by the developers to create an application based on Android platform. This kit contains some sample projects with their source code, their development tools, libraries and emulator so that a developer can create and test the application at same time. As all android applications are written in java language and which are run on predefined Dalvik machine which is particularly a virtual machine which is basically designed for running application on top of Kernel. Android SDK contains various tools and platforms with various other packages which can be downloaded using SDK manager. Therefore whenever any new version is arrived or any other update is released user can update it by downloading them.

3.6.2 SQLite:

SQLite is basically an open source SQL database which stores its data in text file on device. Android has built in SQLite database implementation. As SQLite supports relational database feature and also we does need any kind of connection establishment as what we do with other databases for connecting with database. SQLite performs multitasking and can do read and write at the same time and can store entire database as single cross platform files on host machines.

Code for the SQLite is basically in public domain and therefore it is open to use for any purpose as it may be commercial or private. As SQLite is very compact library, it requires limited memory during run time 250 kb. Therefore performance wise it can even run in low memory environment too. SQLite supports data types such as TEXT, INTEGER, REAL.

3.7 Proposed Flowchart for Android security framework

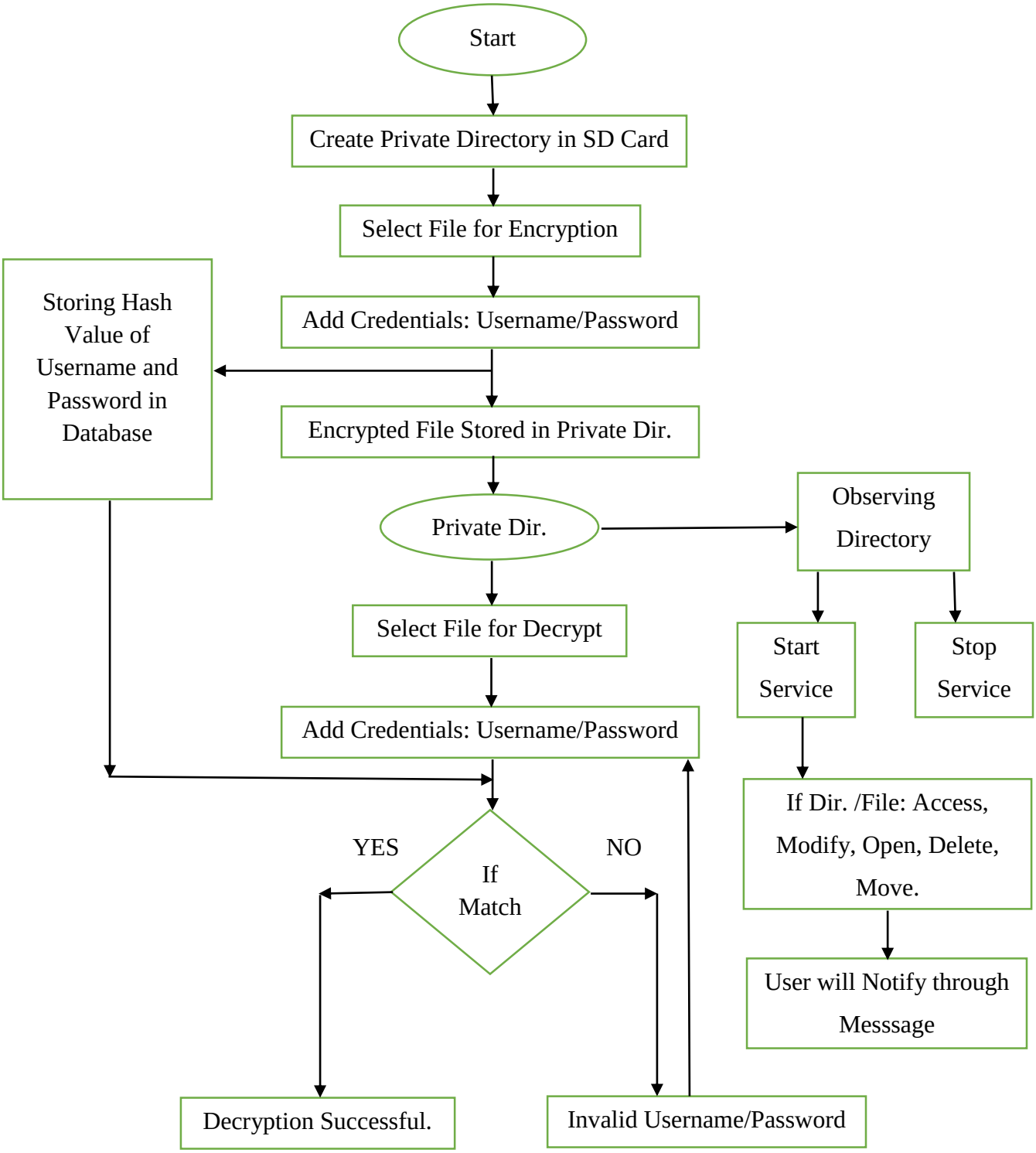

Figure 5: Flow chart

3.8 Proposed steps:

Step1. Creating Private directory to store encrypted files.

Step2. Select particular file to Encrypt from list of files in SD card.

Step3. User has to provide Credentials for particular file. User's credentials as username and password are hashed and stored in Database. And stored hash digest and salt is checked with newly created digest value while authentication. If it is matched with stored value, user gets access to file. Thus validating the user every time can grant access to those encrypted files.

Step4. Encrypted files are stored in Private directory.

Step5. Select file to Decrypt by providing same Credentials while given at Encryption.

Step6. Hash of the Credential values are matched against the saved hash value from Database.

If Hash is matched, then Decryption of File is carry forward.

Else user has to provide Credentials again.

Step7. Start service will run a service class which will observe the Private directory and it's File.

If Directory/File is Access, Open, Modify, Delete, Move, and then user will notify through message.

Chapter 4

RESUTS AND DISCUSSION

In this chapter, we are describing creation, designing and testing of our application. In this section, we will be describing the actual working of our application with particular snapshots at each step.

Snapshots:

a) Emulator start window, showing the Front screen before Application starts.

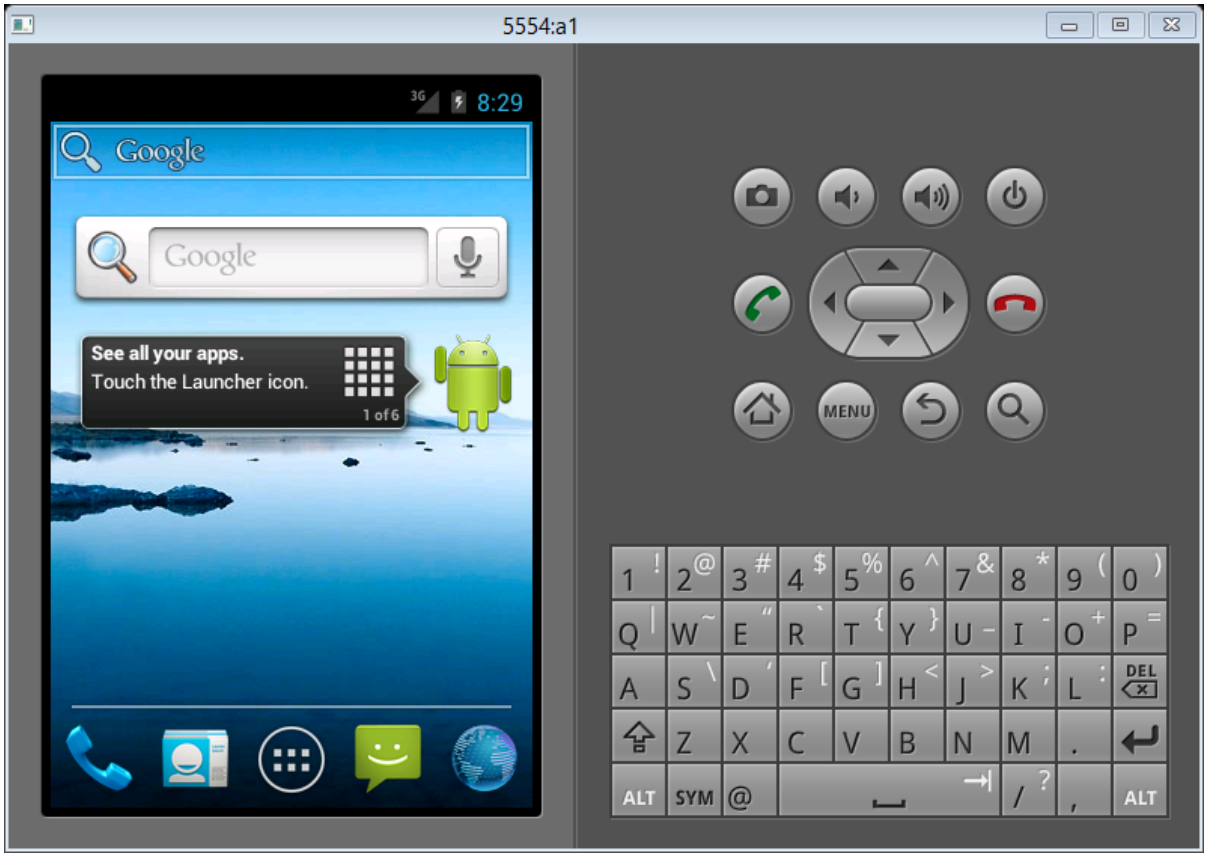

Figure 6: Showing Front Screen of Emulator.

b) Emulator showing SD-card contents after Application starts. List consist of various Folders and Files preloaded in SD-card. Main screen also shows the particular path of SD-card loaded e.g. /sdcard.

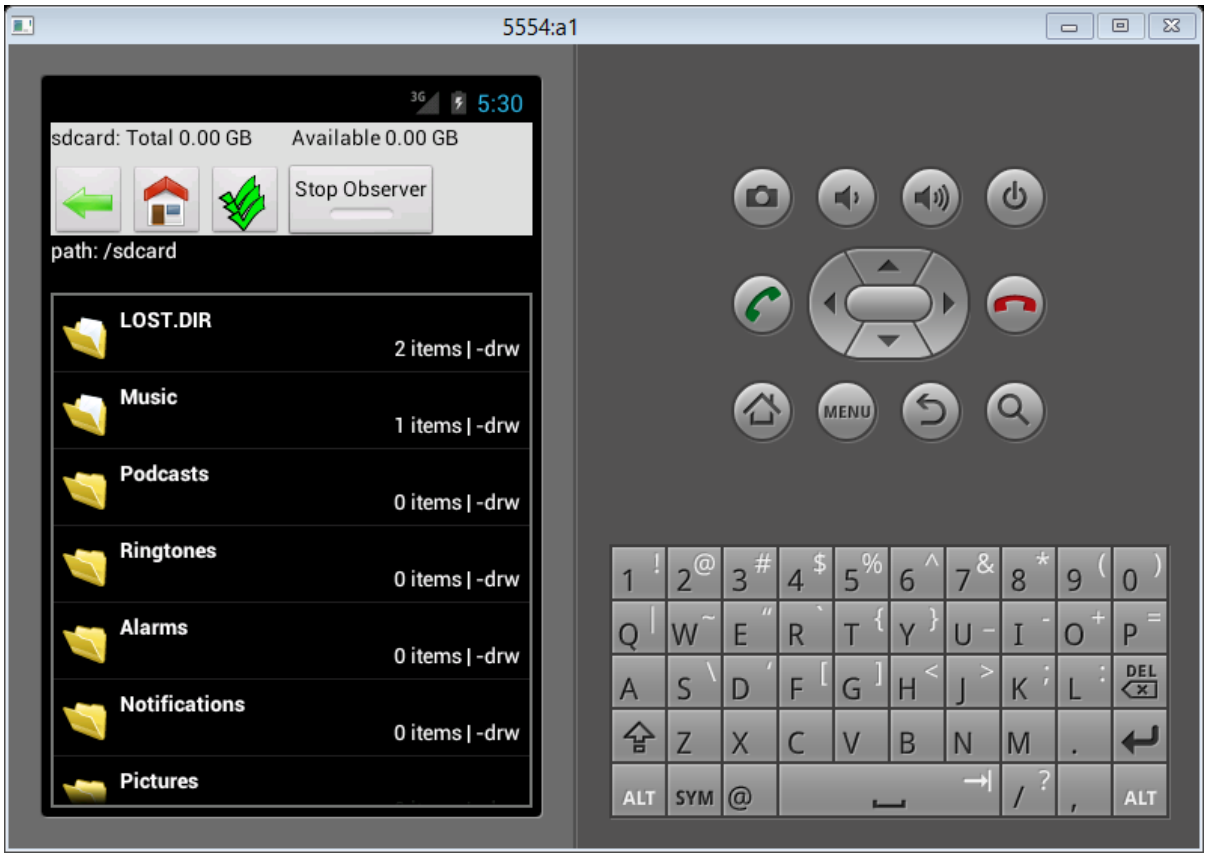

Figure 7: Listing SD card directory

c) Emulator showing /Download Directory in SD-card. Which consist of various files as Plaintext.txt, Article.docx, piuva.jpg, Forest.jpg, drops.jpg and a Punjabi song Yaari Chandigarh Waliye.mp3 with their respective size and permission.

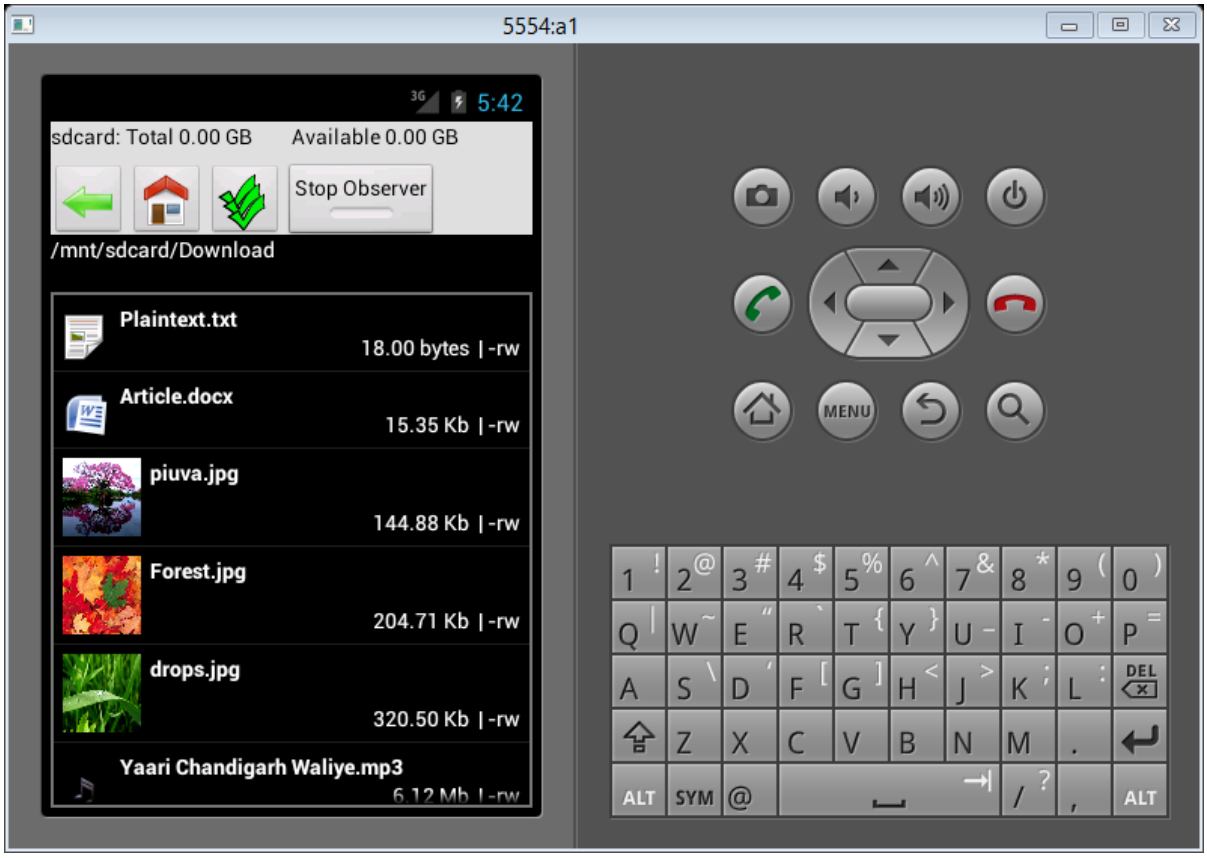

Figure 8: Showing contents of /Download directory

d) Suppose Plaintext.txt file is selected for Encryption from the listed files from /Download directory of SD-card. The file will be choose by holding long press on particular file.

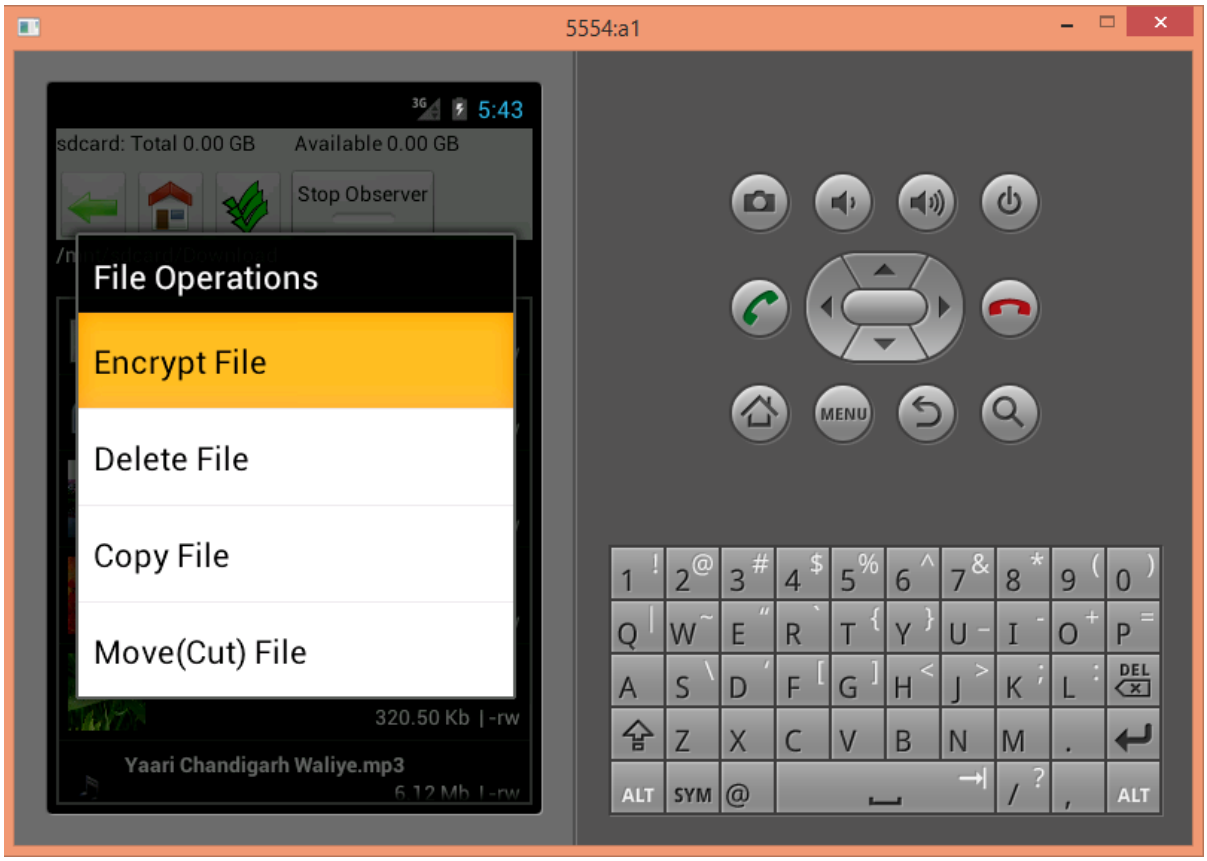

Figure 9: Selecting fie for Encrypting

e) After selecting the Encrypt text in selection, new small window will be created. This small window consist of username and Password as user's Credentials which will be store and use for validating user later on.

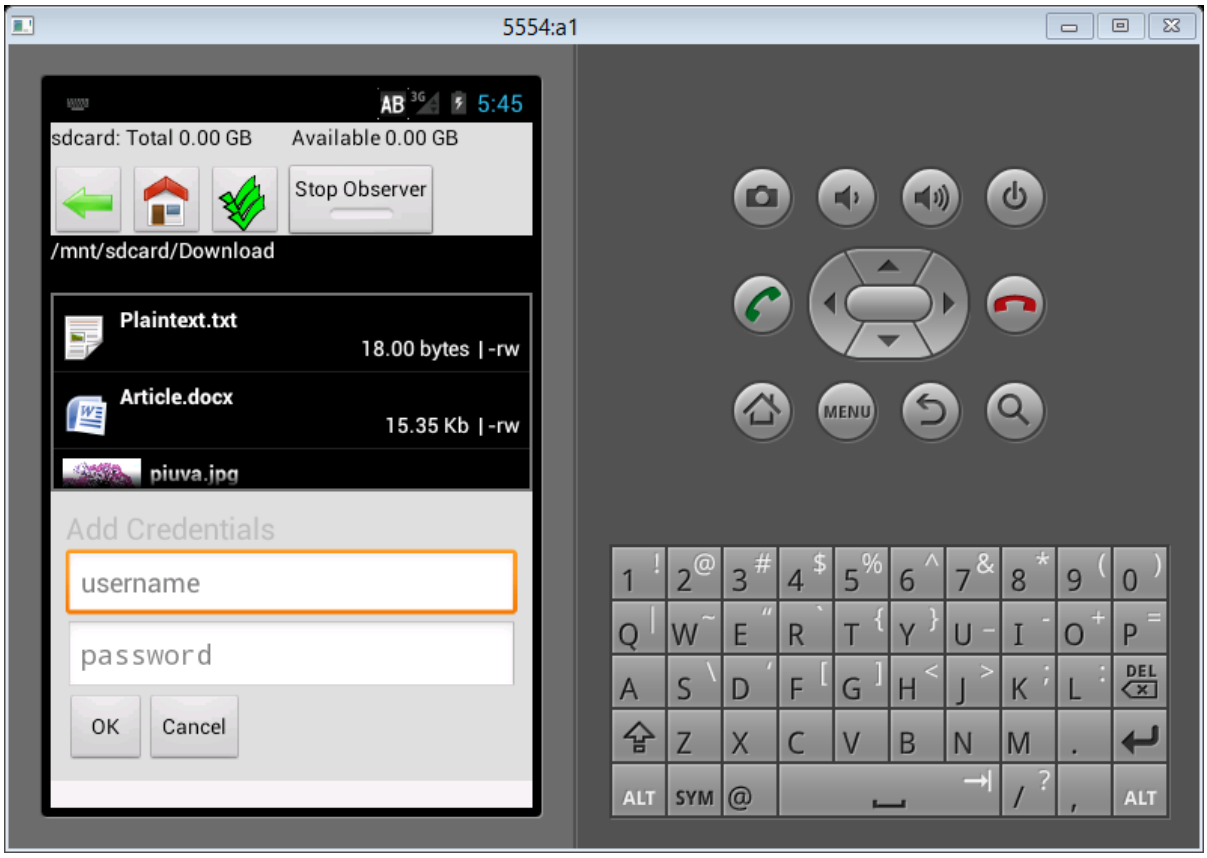

Figure 10: Credential window is created

f) Validation on filling the Credentials are applied, both the fields cannot be set to null. Therefore user has to input in both fields to remove the alert.

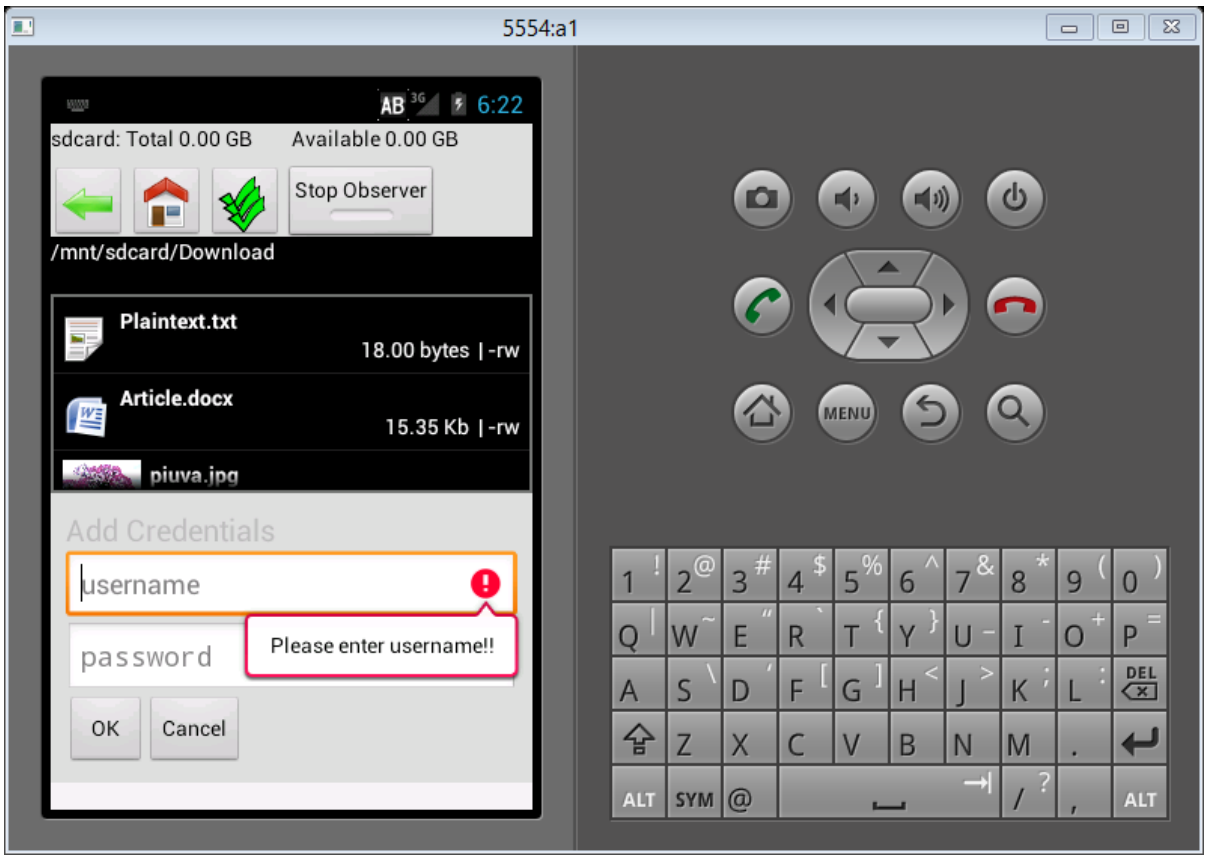

Figure 11: Showing Validation on Username

g) After adding Credentials by User the values are hashed and store in Database. These hash values are used for validating user each time it will decrypt the file.

These hash values with filename is stored in Database at backend of application.

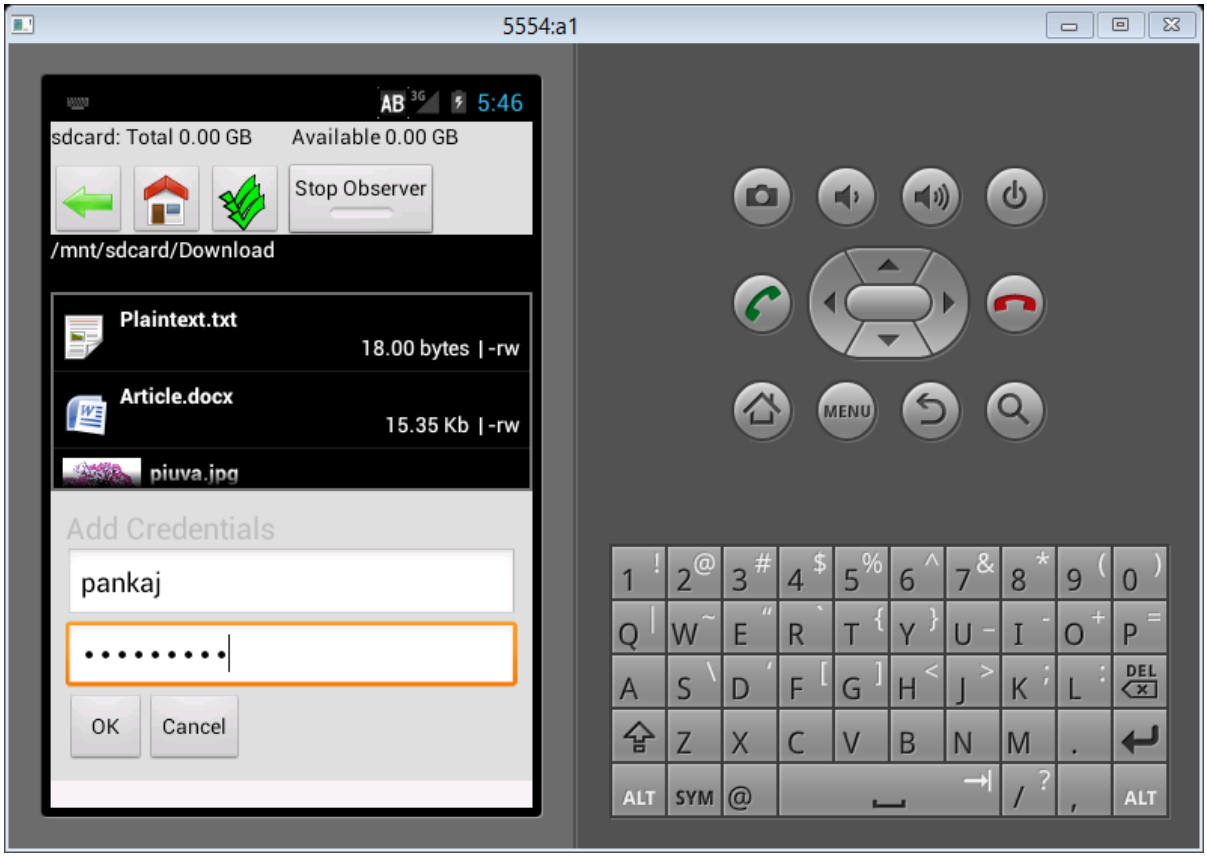

Figure 12: Adding Credentials to File

h) After adding the Credentials, Encryption process starts and bit by bit value is encrypted and new encrypted file is created.

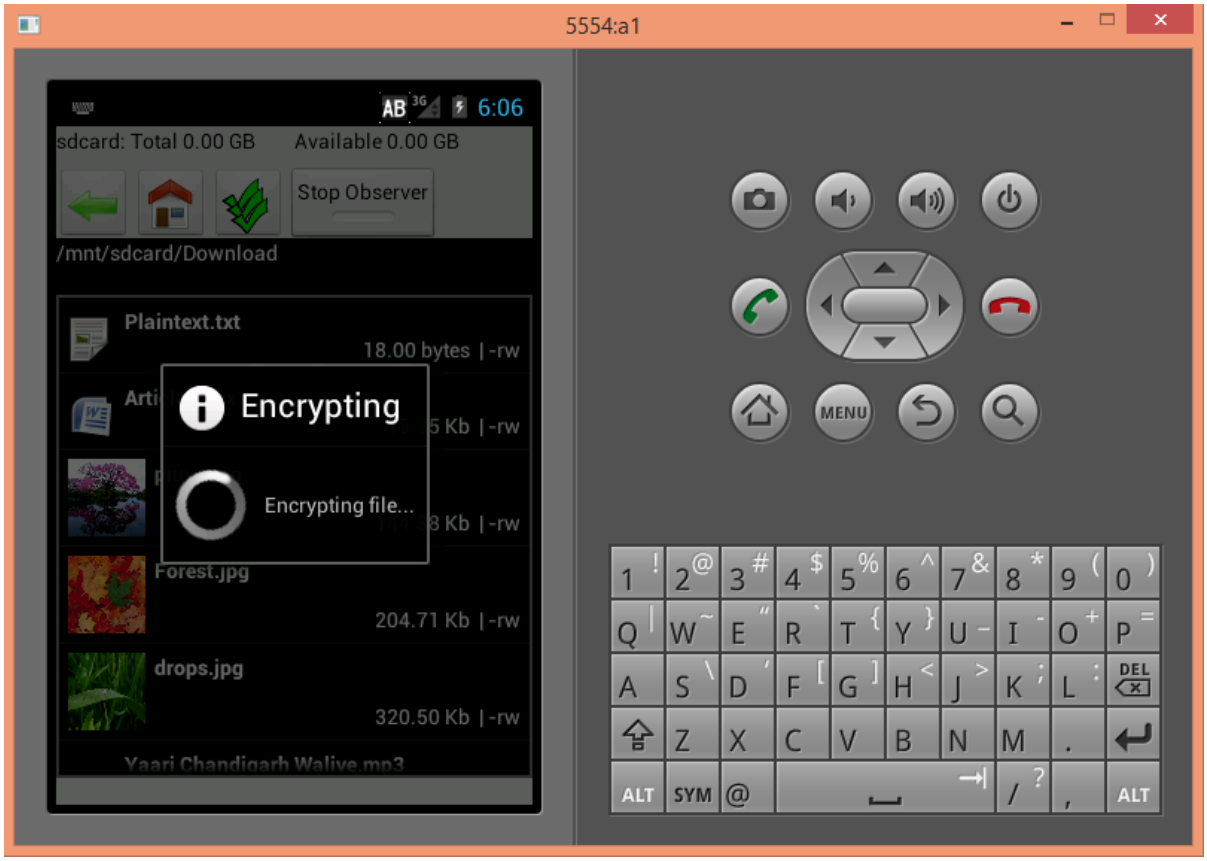

Figure 13: Encryption in Progress

i) After successfully encrypting a file, message is generated “Encrypted Successfully” with showing path of the stored encrypted file which is in this case in Private directory /sdcard/Filesecure and original file is removed automatically.

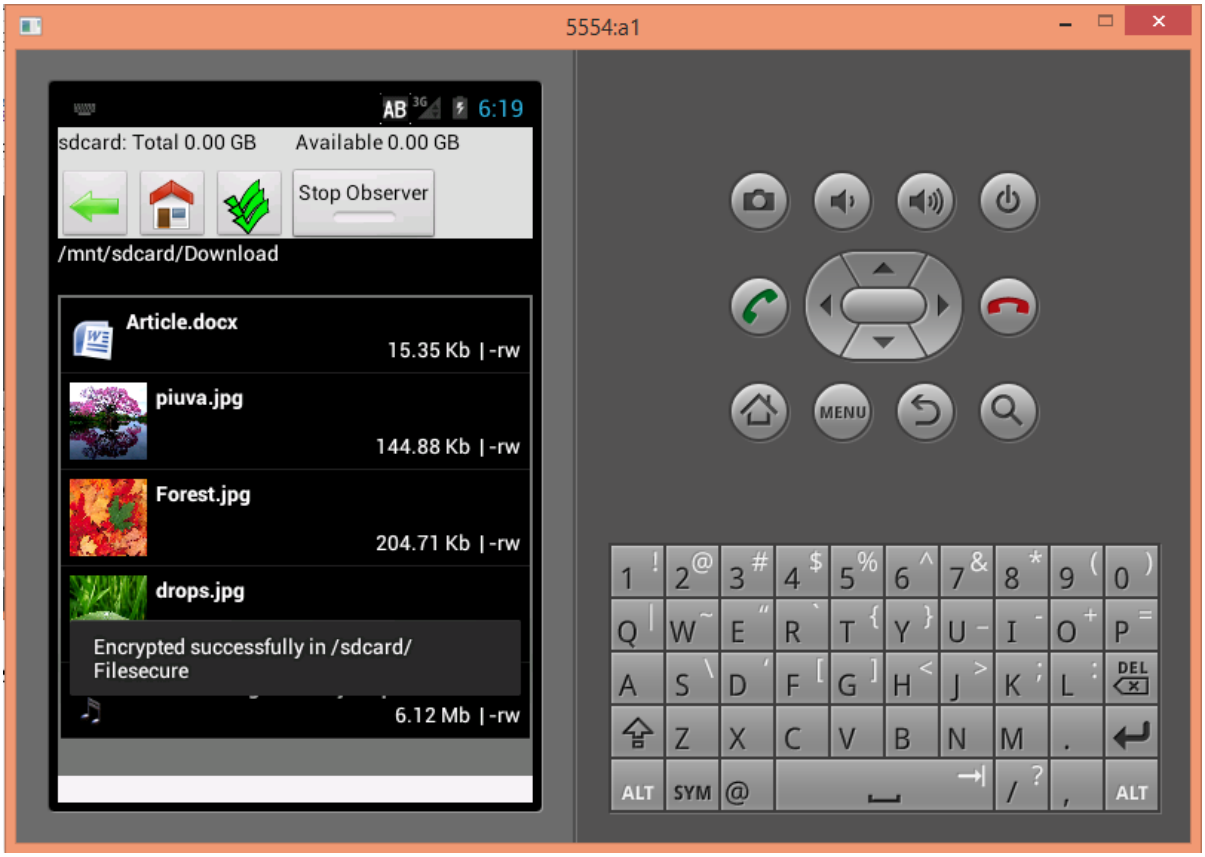

Figure 14: Storing encrypted file in /sdcard/Filesecure directory

j) Emulator showing the content of Private Directory, consist of Encrypted file stored with extension .enc. with size of file and permission. As Private Directory is /sdcard/Filesecure.

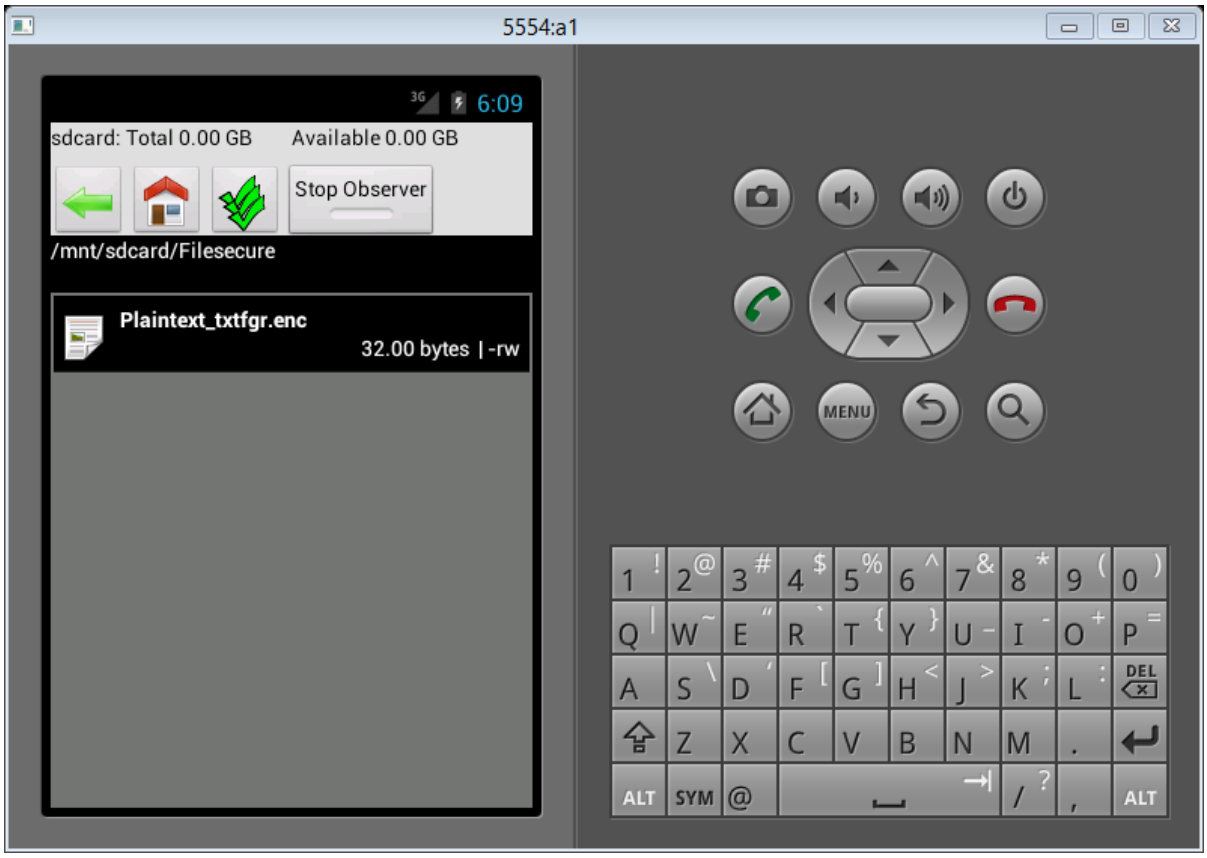

Figure 15: Showing encrypted file

k) Suppose user wants original file from Encrypted file. After long pressing the encrypted file, Decrypt file option is generated.

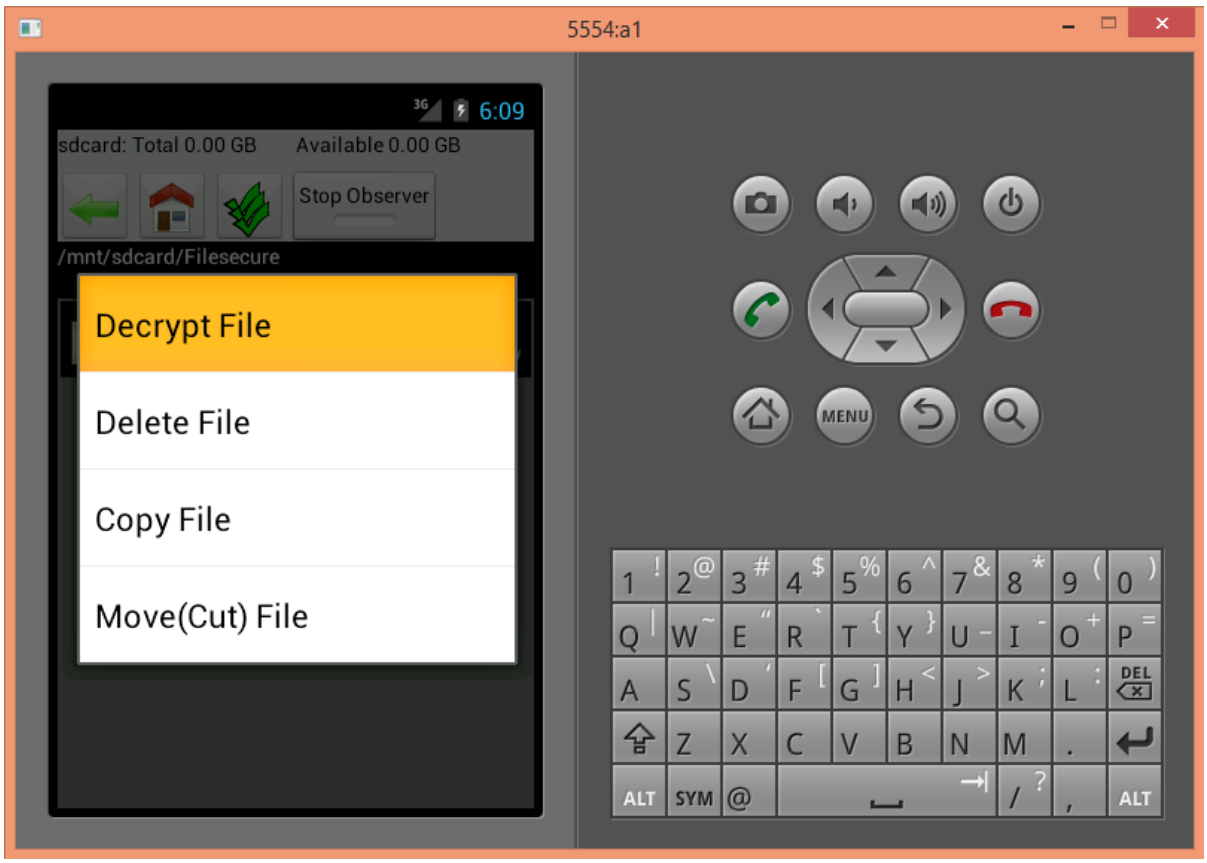

Figure 16: Selecting file for decrypting

l) Now user has to provide same Values of Username and Password which is used during encryption.

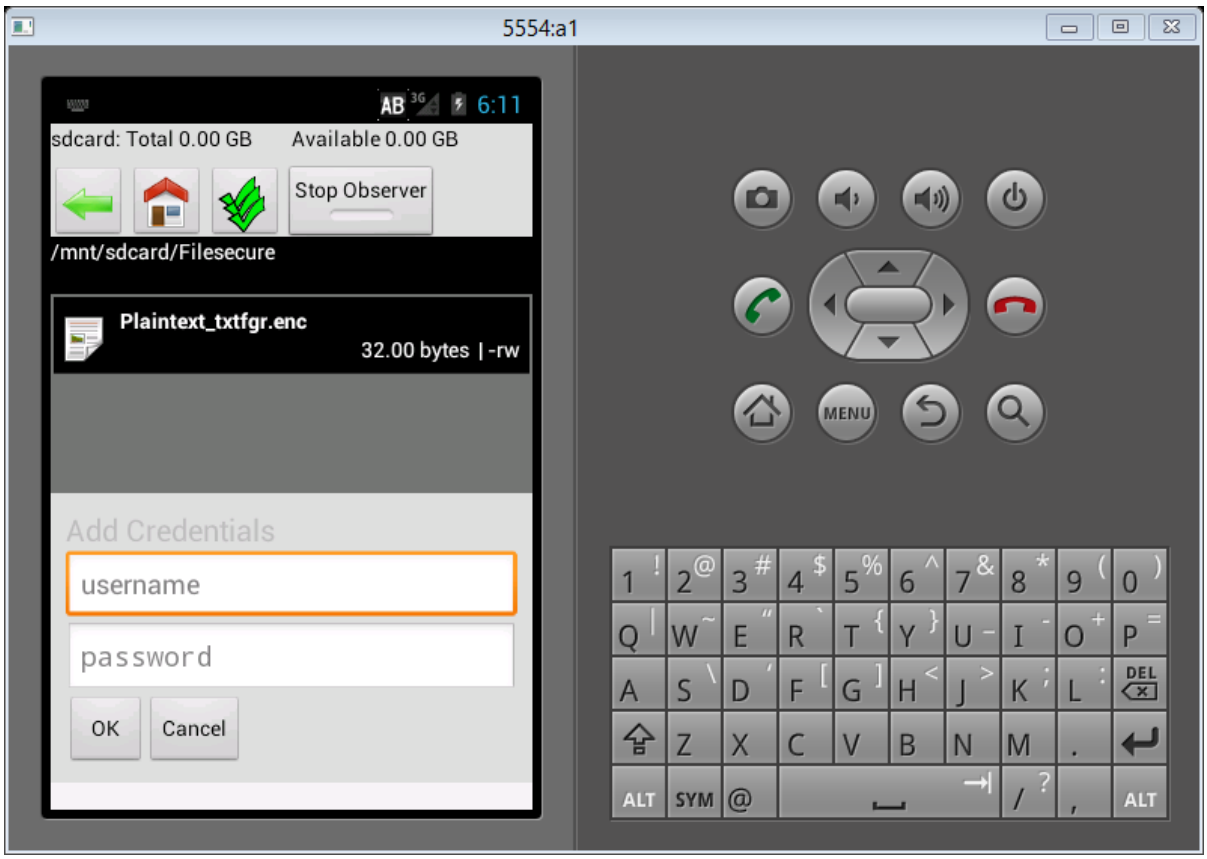

Figure 17: Showing Credential window at Decryption end.

m) Matching the Credentials with stored values in database. If it does not match error message is generated saying “Invalid username/password”

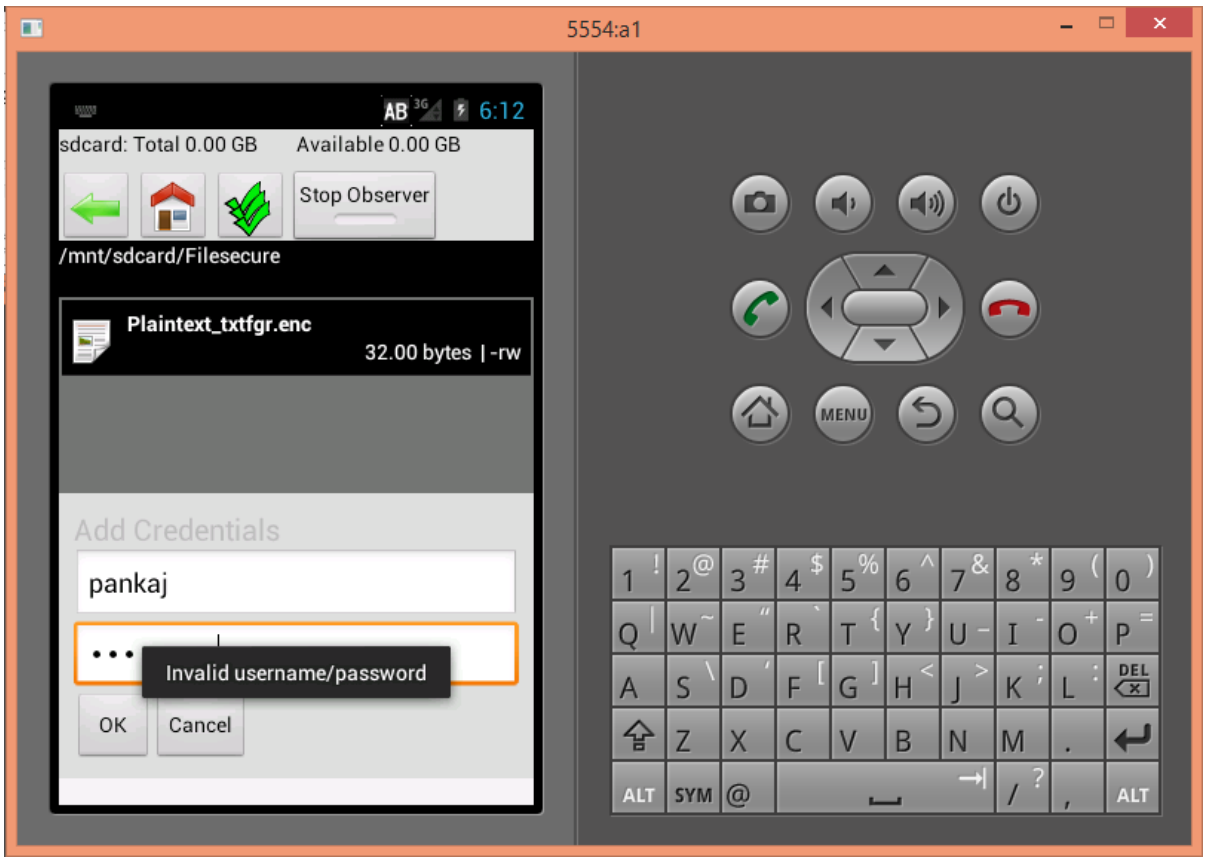

Figure 18: Validating the hash value from SQLite database

n) Matching the Credentials with stored values in database. If it does not match error message is generated saying “Invalid username/password”

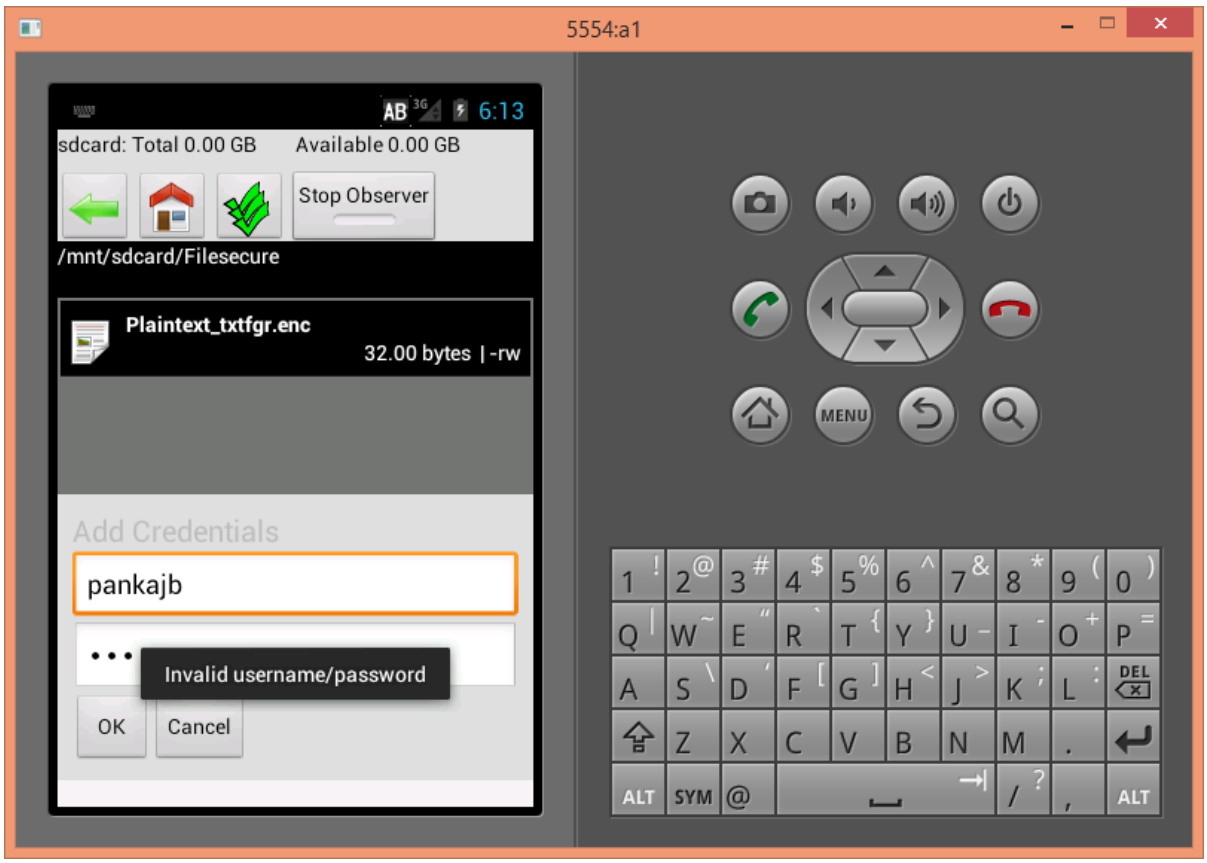

Figure 19: Validating the user credential from SQLite database

o) If Hash values are matched, Decryption of file will carry forward. Emulator showing decryption process storing the original file in directory.

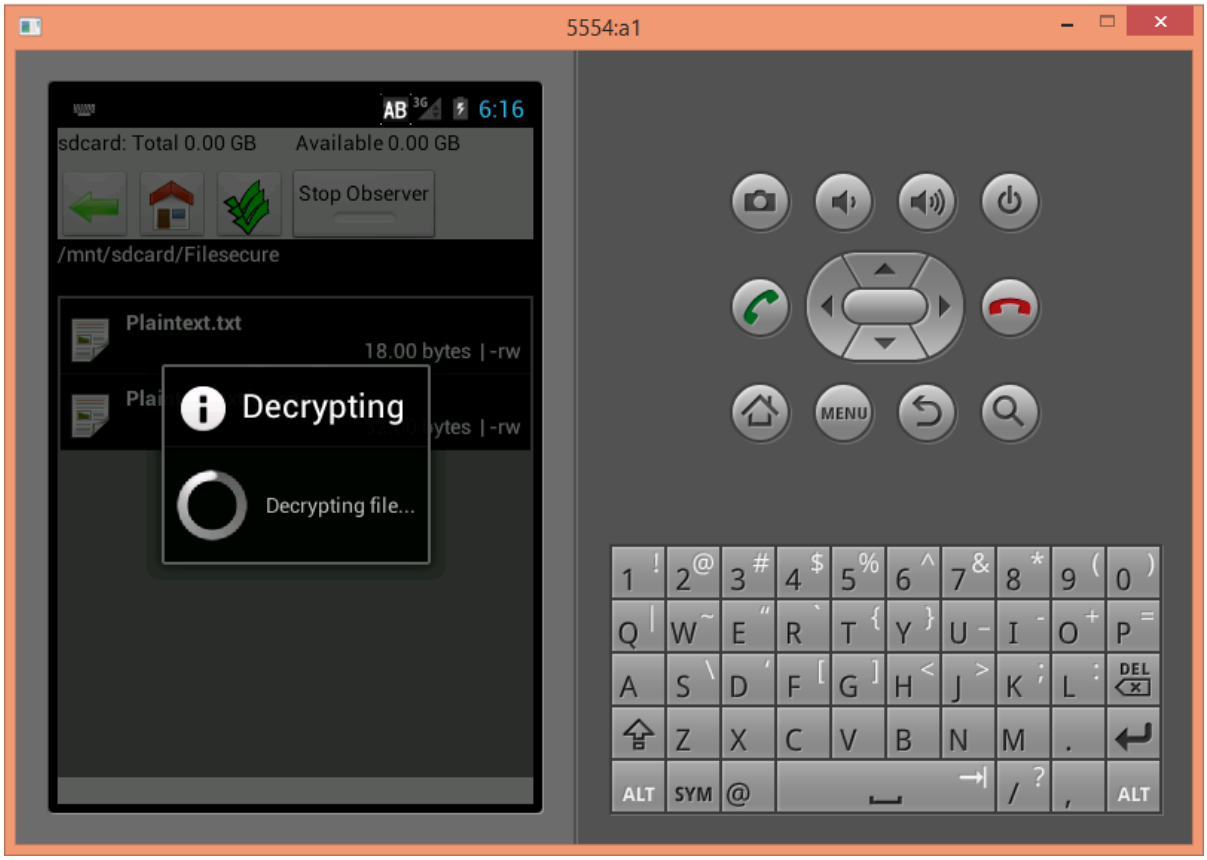

Figure 20: Showing Decryption in Process

p) After successfully Decrypting a file, message is generated “Decrypted Successfully”. Where original file Plaintext.txt is store in same directory.

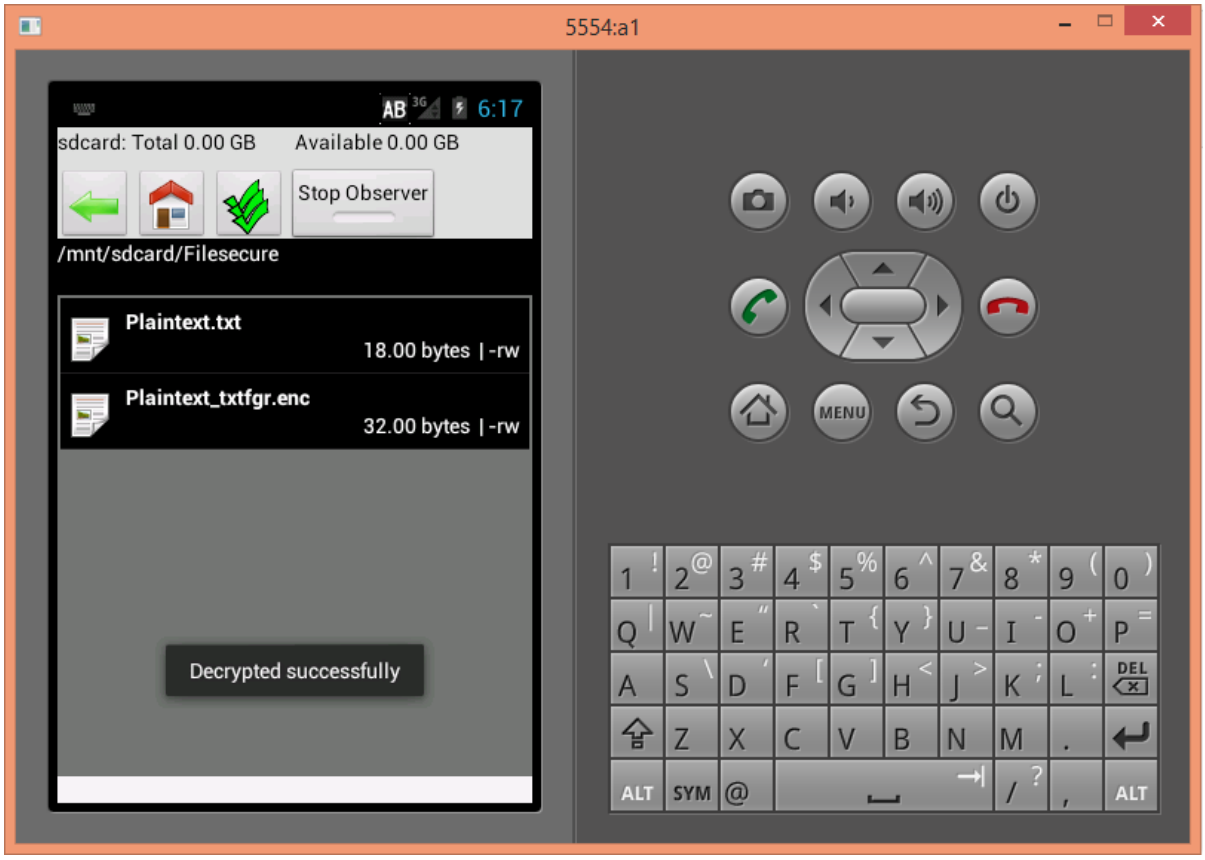

Figure 21: Showing Decryption successfully

q) Service is start by pressing the Toggle but in ON state, it is for monitoring the Private directory and its files. The service will detect if any other process of other application trying to access the files.

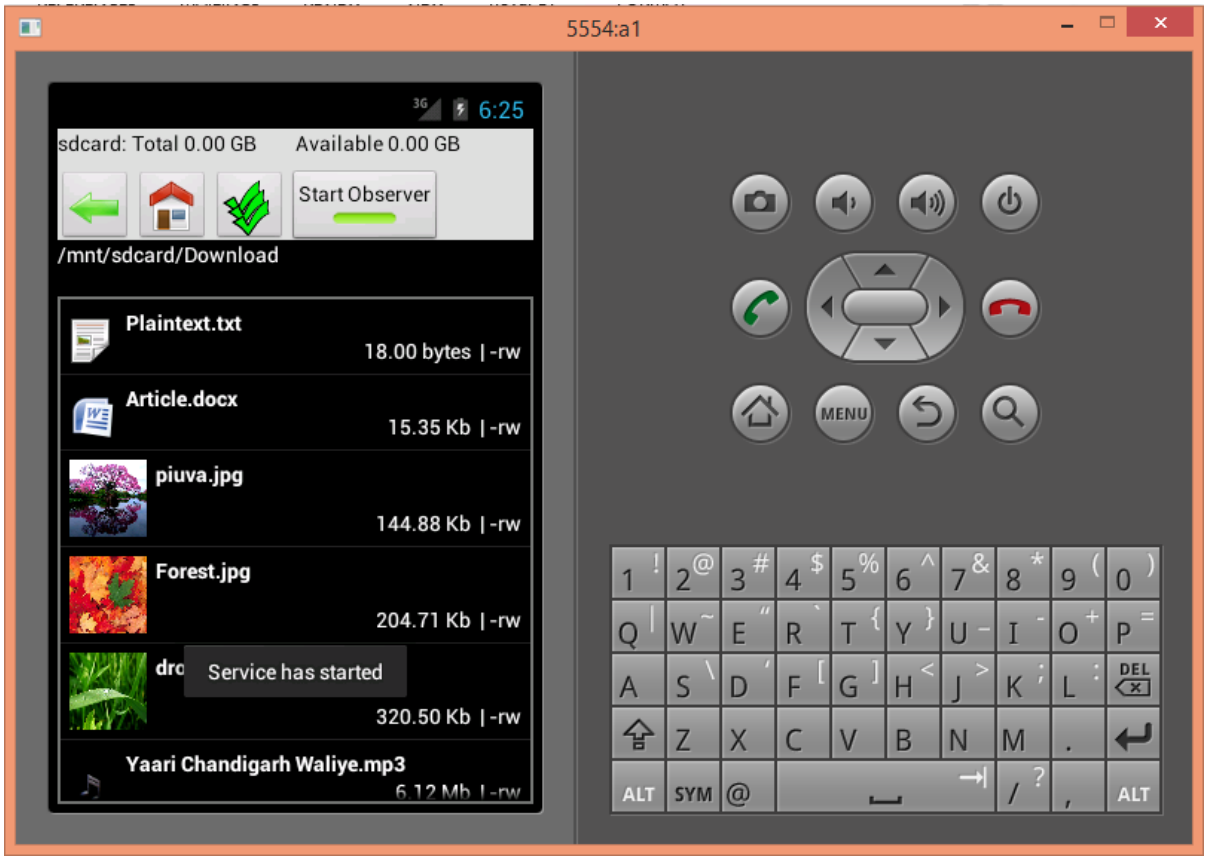

Figure 22: Showing Service start successfully

r) As service shows the message whenever any process tries to read Private Directory contents or trying to open it.

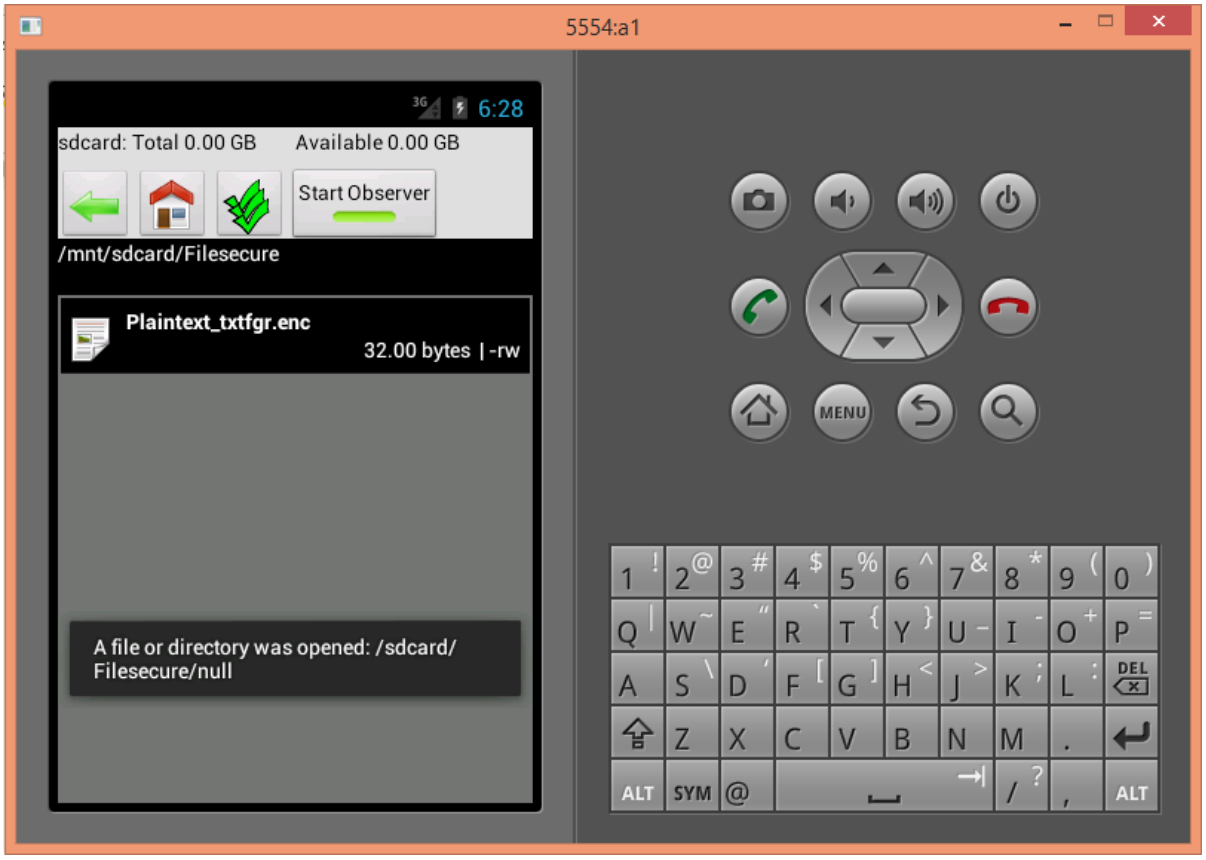

Figure 23: Showing message, Private directory was opened

s) Message has been pop up when encrypted file is accessed or read.

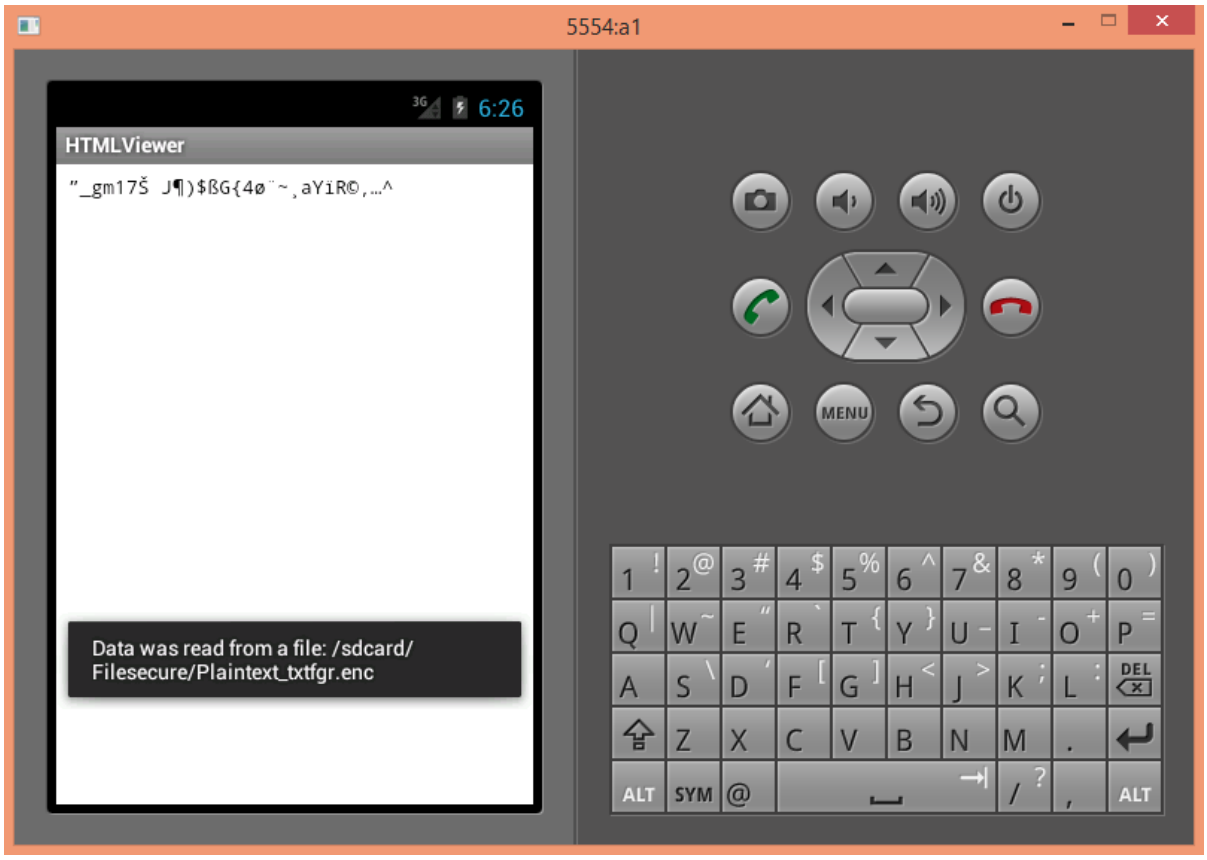

Figure 24: Showing message, encrypted file has accessed

t) By pressing the Toggle button to OF state, service will stop and application will no longer detect or monitor the Private directory and its files.

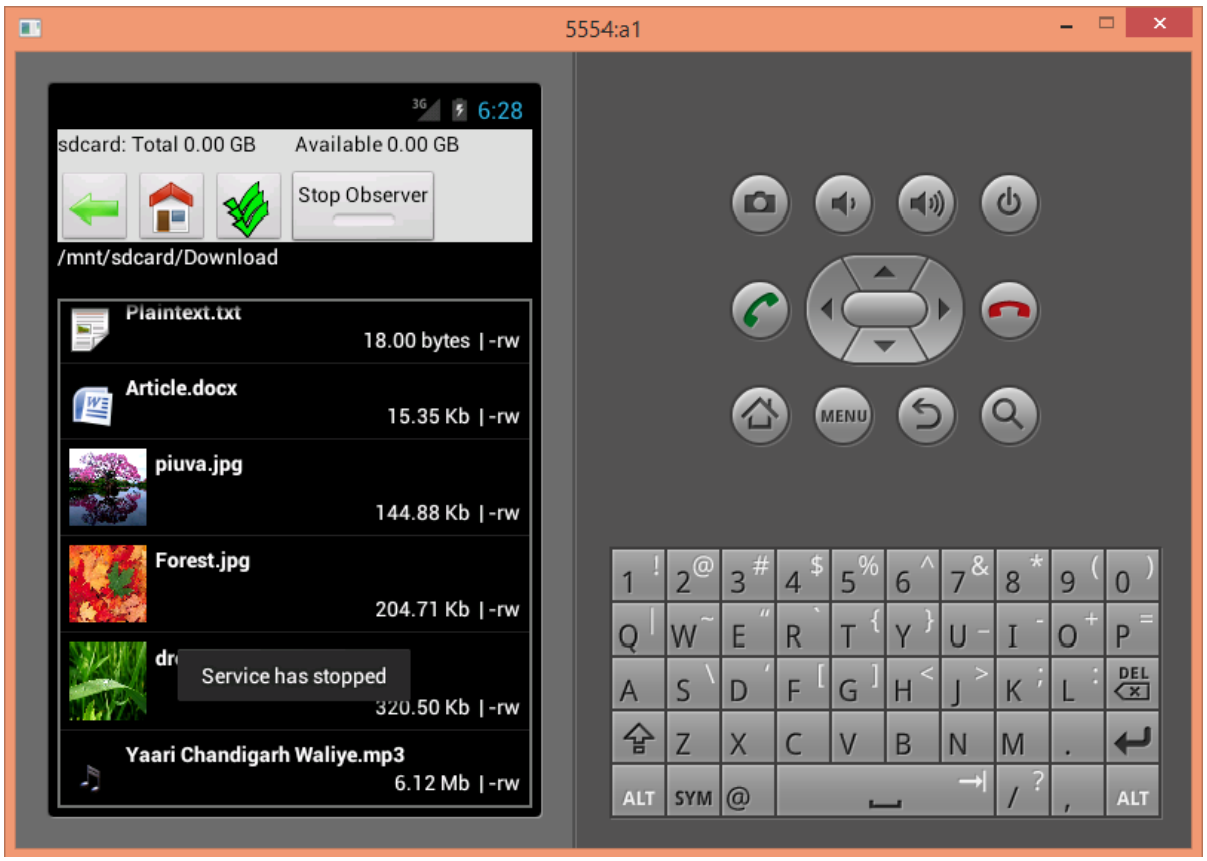

Figure 25: Showing Service stop message

u) This snapshot shows the Database table, consist of username with their stored Hash values of password and salt. Table also consist of name of encrypted file.

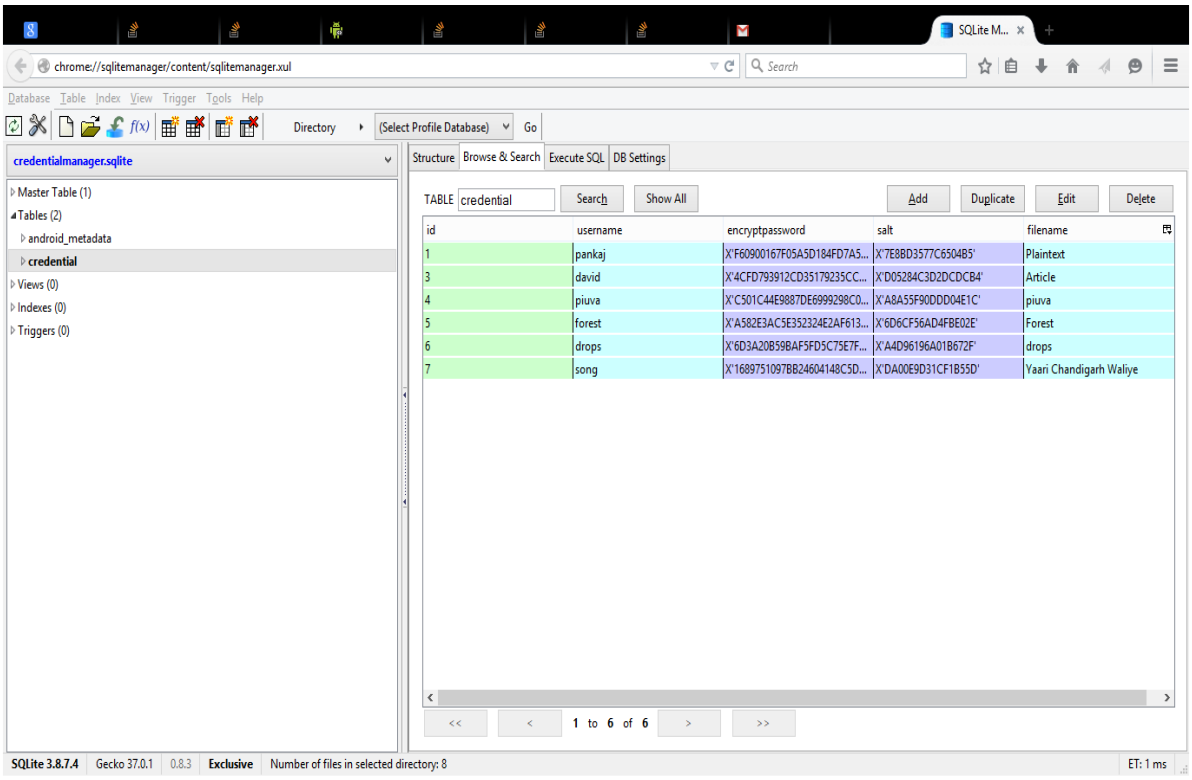

Figure 26: Showing Content of SQLite database

Chapter 5

CONCLUSION AND FUTURE SCOPE

From latest statistics Android has become world's most popular mobile operating system. As with increase in popularity day by day, it also vulnerable to various malicious applications. Android provides some very strong feature to application as application sandboxing, inter-process communication, with strong permission model, cryptography and file system isolation. Inappropriately, as all of these protections could be bypass in situation like in case rooting of system. To get rid from this type of data disclosure situation one approach of data protection technique is cryptography. A further detection with prevention of processes accessing the Private Directory will be act as future scope.

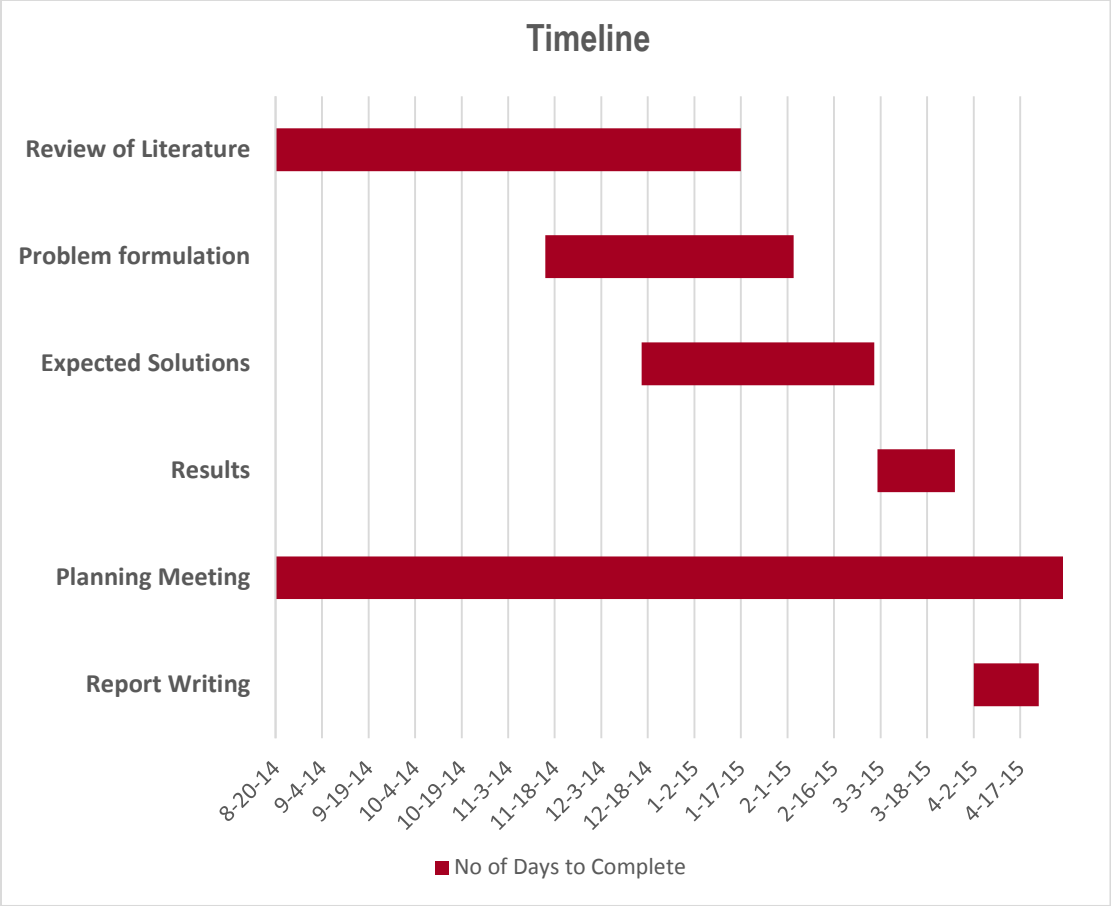

Figure 7. Timeline

Chapter 6

LIST OF REFERENCES

I. Research Papers

[1] S. C. Lauren Darcey, A Brief History of Mobile Software Development, 2009.

[2] david, "Android and its versions," 16 july 2014. [Online]. Available: http://tech-updates4u.com/. [Accessed 20 nov 2014].

[3] Smieh, "Anatomy Physiology of an Android," 2012.

[4] GADI007, "Android Architecture and Pen Testing of android application," 11 feb 2013. [Online]. Available: http://resources.infosecinstitute.com/android-architecture-and-pen-testing-of-android-applications/. [Accessed 20 nov 2014].

[5] Raluca-Alina, "Protecting Stored Data on Android Devices," *Journal of Mobile, Embedded and Distributed Systems,* vol. IV, p. 2067 – 4074, 2014.

[6] A. N. W. E. A.-R. S. Stephan Heuser, "ASM: A Programmable Interface for Extending Android Security," 2014.

[7] J. P. Muneer Ahmad Dar, "A Novel Strategy to Enhance the Android," *International Journal of Computer Applications,* vol. 91, pp. 0975-8887, 2014.

[8] D. sukhija, "A Review Paper on AES and DES Cryptographic," *International Journal of Electronics and Computer Science Engineering,* vol. III, no. 4, pp. 354-359.

[9] J. K.-j. C. C. P. Chanhee Lee, "Implementing and Optimizing an Encryption Filesystem on Android," in *IEEE*, 2013.

[10] M. A. K. M. A. K. Y. A. Z. Hammad Banuri, "An Android runtime security policy enforcement framework," p. 631–641, 2012.

[11] R. M. A. S. Zhaohui Wang, "Implementing and Optimizing an Encryption Filesystem on Android," in *IEEE*, 2012.

[12] S. U. P. P. Rohan Rayarikar, "SMS Encryption using AES Algorithm on Android," *International Journal of Computer Applications,* vol. 50, no. 19, p. 0975 – 8887, 2012 .

[13] A. S. a. M. Mannan, "Mobiflage:Deniable Storage Encryption for Mobile Devices," San Diego, CA, USA, 2013.

[14] X. J. Yajin Zhou, "Dissecting Android Malware: Characterization and Evolution," in *IEEE Symposium on Security and Privacy*, 2012.

[15] Synetech, 14 July 2012. [Online]. Available: http://superuser.com/questions/448965/does-full-disk-encryption-on-ssd-drive-reduce-its-lifetime.

II. Books

S. C. Lauren Darcey, A Brief History of Mobile Software Development, 2009.

III. Reports and Official Documents

Smieh, "Anatomy Physiology of an Android," 2012.

IV. Websites

http://resources.infosecinstitute.com/android-architecture-and-pen-testing-of-android-applications/.

http://tech-updates4u.com/.

http://superuser.com/questions/448965/does-full-disk-encryption-on-ssd-drive-reduce-its-lifetime

Chapter 7

APPENDIX

ABBREVIATIONS

a) OHA – Open Handset Alliance

b) VM - Virtual Machine

c) UID – Unique Identity

d) GID – Group Identity

e) SHA – Secure Hash Algorithm

f) AES – Advance Encryption Standard

g) NIST – National Institute of Standard and Technology

Chapter 8

PUBLICATIONS

Pankaj Laheri, Anu Garg, "A SURVEY ON EXISTING ANDROID PERMISSION BASE MODEL", International Journal of Applied Engineering Research (ISSN: 0973-4562)