Encryption And Decryption Of

Data On Images

INDEX

Abstract

1. Introduction

1.1. Existing System with Limitations

1.2. Proposed System with features

2. Feasibility Report

2.1 Economical Feasibility

2.2 Operational Feasibility

2.3 Technical Feasibility

3. Requirements Specification

3.1 Hardware Requirements

3.2 Software Requirements

4. Behavioral Description

4.1 Data Flow.

4.1.1 Data Flow Diagrams

4.1.2 Use-Case Documentation

4.2 Process Flow

4.2.1 Activity Diagrams

5. Functional Description

6. Performance Requirements

7. Design

7.1 Sequence Diagram

7.2 Collaboration Diagram

7.3 State Chart Diagram

9

10

11

12

12

13

14

14

15

17

23

26

41

42

43

45

49

Page 5

7.4 Use case Design

7.5 Class Diagram

7.6 Activity diagram

8. Coding

9. System Testing and Implementation

10. Output Screens

11. Conclusion

Bibliography

51

52

54

56

61

67

75

76

Page 6

LIST OF DIAGRAMS

1. SEQUENCE DIAGRAM:

USER

ADMIN

2. COLLABORATION DIAGRAM

3. STATE CHART DIAGRAM:

ADMIN

USER

4. USE CASE DIAGRAM

5. CLASS DIAGRAM

6. ACTIVITY DIAGRAM:

ADMIN

USER

46

47

48

49

50

51

53

55

56

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 7

LIST OF OUTPUT SCREENS

SCREEN 1: HOME PAGE

SCREEN 2: ABOUT US PAGE

SCREEN 3: LOGIN PAGE

SCREEN 4: REGISTRATION PAGE

SCREEN 5: DOWNLOAD PAGE

SCREEN 6: UPLOAD PAGE

SCREEN 7: CHANGING PASSWORD PAGE

67

68

69

70

71

72

73

LIST OF TABLES

TABLE 1: LOGIN TABLE

TABLE 2: REGISTRATION TABLE

TABLE 3: UPLOAD TABLE

74

74

74

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 8

**ABSTRACT**

Android is a software platform and operating system for mobile devices. This is a project that is used to perform secret data transmission by performing encryption of text on images. The sender uses a key to perform encryption and the same key is given to the receiver to decrypt and obtain the data.

***INTRODUCTION***

Encryption of data on images provides a safe and secure transmission of data between the sending and receiving party.

The data/text which the sender wants to transmit is selected first and then an image is chosen from the current mobile device.

The chosen text is then encrypted in the image such that the data is not visible to any third party.

After encryption is performed, the image is sent to receiving party and the receiver decrypts the data using the key given by the sender.

This decryption process can only be performed by an authenticated receiver using this application.

***OBJECTIVE***

Main objective of this application is to provide a secure and secret transmission of text by encrypting it on an image using a key and which can only be decrypted by by an authenticated receiver using the same key on the same application.

***KEY FUNCTIONALITY***

* Start
* Chosen text/retrieve data from current device
* Browse an image from the device

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 9



* Encrypt the chosen text onto the image
* This image is sent to the receiver
* Receiver now decrypts the data

***SOFTWARE AND HARDWARE REQUIREMENTS:***

Java, JDK, Android SDK, My Eclipse

Android Mobile Device

**1. INTRODUCTION**

1.1. Existing system with limitations

The perspective of data security, which has always been an important aspect of

quality of service, Cloud Computing inevitably, poses new challenging security threats fornumber of reasons.

Encryption of data/text was previously done using hash and cryptographic algorithms, which is written in binary form. Such encryption techniques are visible in a specific form and the intruder who is aware of binary text formats can easily read the text.

In those times it provided a lot of security but as technology progressed many hackers also came into existence and it could not provide security upto that extent.

1.2. Proposed system with features:

In this project we used many techniques to encrypt and make the data in-visible to any one(not even to the receiver). We used LSB (Least Significant Bit) format, stenography and cryptographic techniques to encrypt the data and the data/text being encrypted on an image will not be visible.

.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 10

The sender will use a key for security reasons and the same key should be used by receiving party to decrypt the image and obtain the text written on it.

The image on which data encryption is being performed can be chosen from the existing device and after all the process completes, that image can be sent through Bluetooth or mail or google drive e.t.c.,



2. FEASIBILITY STUDY

Feasibility study is an important phase in the software development process. It

enables the developer to have an assessment of the product being developed. It refers to thefeasibility study of the product in terms of outcomes of the product, operational use andtechnical support required for implementing it.

Feasibility study should be performed on the basis of various criteria and parameters. Thevarious feasibility studies are:

Economic feasibility

Operational feasibility

Technical feasibility

2.1. Economic feasibility:

An economic feasibility test focuses on returns and costs of a proposed plan in boththe short and long-term. An economic feasibility study (EFS) should consider investment andoperating costs, the time value of money, risk and uncertainty, quality of available data, andthe sensitivity of assumptions. An economic feasibility study should demonstrate the netbenefit of the proposed course of action in the context of direct and indirect benefits and coststo the organization and to the general public as a whole

Our project reduces the expenses in the manual system because Our project is

applicable for only internet. So, it is economically feasible

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 11

2.2. Operational feasibility:

Issues that appear to be relatively minor in the beginning have ways of growing into

major problems after implementation. Therefore, all operational aspects must be considered

carefully. Operational feasibility means the project should be supported by users. It should

not cause any problems to users after implementation.

Our project can be implemented and executed in any type of networking environment.

So, it is operational feasible.

2.3. Technical feasibility:

Technical feasibility is nothing but implementing the project with existing technology.

In the technical feasibility the following issues are taken into consideration

1. Whether the required technology available or not

2. Whether the required resources are available

3. Software and hardware.

Inour project supports technical feasibility because we implemented our projectusing Microsoft.Net3.5 technology which is platform independent and all the resources thatare required for project are easily available.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 12

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 13

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 14

4. BEHAVIORAL DESCRIPTION

4.1. Data flow:

Data flow diagram indicates the flow of data based on source code. Data flow diagram shows

entire functionality of the system.

There are two basic versions for dataflow diagrams.

Data flow diagram for current system.

Data flow diagram for new or proposed logical system.

Data flow diagram is a well known approach to visualize the data processing in

business analysis field. A data flow diagram is strong in illustrating the relationship of

processes, data stores and external entities in business information system.

1. A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an

information system. It differs from the flow chart as it

data processing (structured design).

2. It is common practice to draw a context level data flow diagram first which shows the

interaction between the system and outside entities. The DFD is designed to show how a

system is divided into smaller portions and to highlight the flow of data between those parts.

This context-level data-flow diagram is then "exploded" to show more detail of the system

being modeled.

3. Data-flow diagrams were invented by Larry Constantine, the original developer of

structured design, based on Martin and Estrus’s “data-flow graph" model of computation.

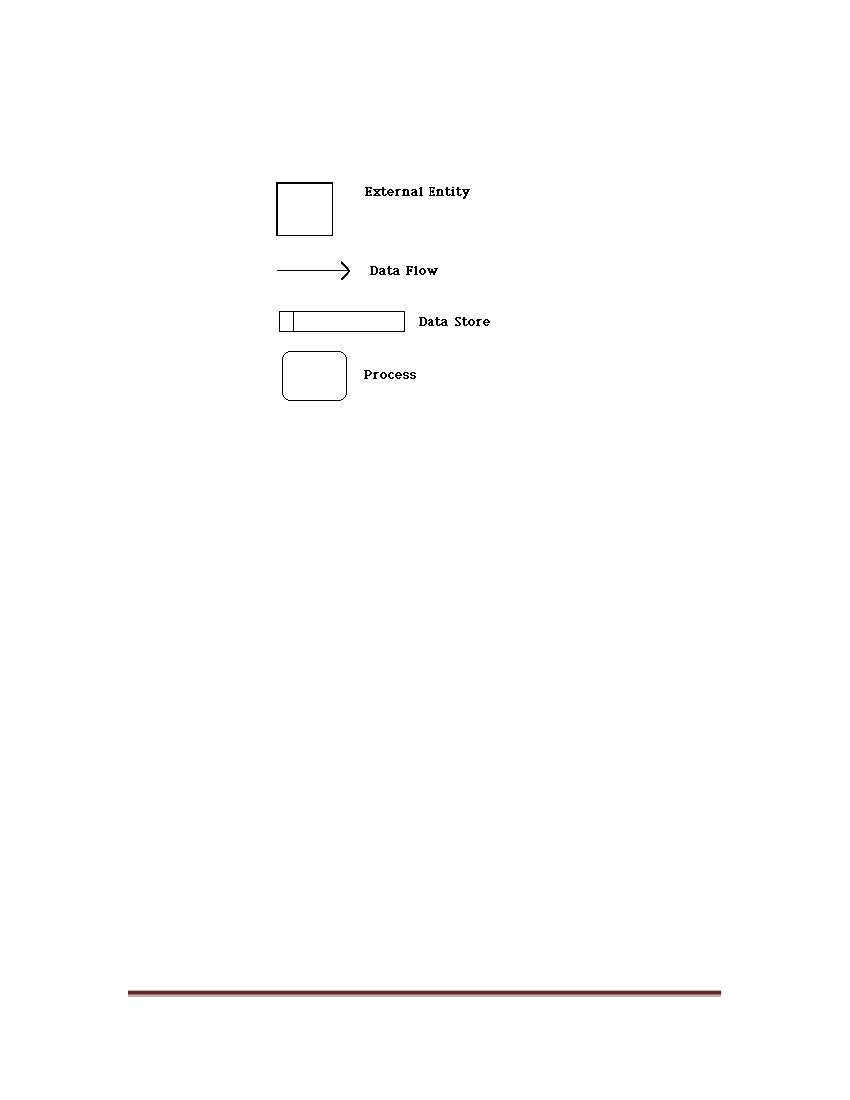
4. Developing a data-flow diagram helps in identifying the transaction data in the data

model.

shows the data flow instead of the

control flow of the program. A data-flow diagram can also be used for the visualization of

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Symbols:

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 16

4.1.1 Data Flow Diagrams:

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

TYPES OF DATA FLOW DIAGRAMS:

DFD’s are of two types

1. Physical DFD
2. Logical DFD

**1.Physical DFD**

Structured analysis states that the current system should be first understand correctly. The physical DFD is the model of the current system and is used to ensure that the current system has been clearly understood. Physical DFDs shows actual devices, departments, and people etc., involved in the current system.

**2. Logical DFD**:

Logical DFD s is the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.

The Basic Notation used to create a DFD s are as follows:

Dataflow: Data move in a specific direction from an origin to a Destination.

**Process** People, procedures, or devices that use or produce

(transform) Data. The physical component is not identified.

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 17

**Source** External sources or destination of data, which may be

people, programs, organizations or other entities.

**Data Store** Here data are stored or referenced by a process in the system.

DATA FLOW DIAGRAMS

Level 0



ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 18

Level 1



ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 19

Level 2:

Sender

Register

Login

Home

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 20

**4.1 Class Diagram**

Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. It is the main building block in object oriented modeling. It is being used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.

The classes in a class diagram represent both the main objects and interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:

* The upper part holds the name of the class.
* The middle part contains the attributes of the class, and
* The bottom part gives the methods or operations the class can take.

In the conceptual design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the statically relations between those objects. With detailed modeling, the classes of the conceptual design are often split in a number of subclasses.

**CLASS DIAGRAM:**

Image encryption()

Send()

Browse()

Show()

Browse()

Getphoto,encrypted key

Image decryption

Photo,code,text

Image encryption

Login()

Registration()

System

Registration()

String name

Registration

Encryption()

String key,text

Decryption()

String key

Text decryption

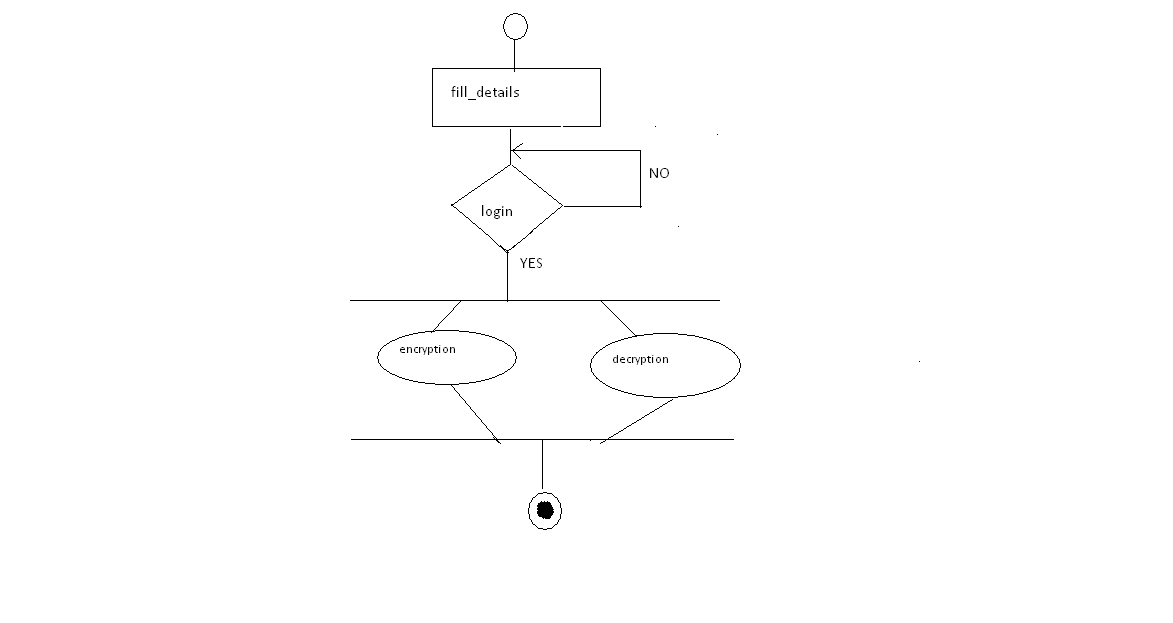
Text Encryption

Login()

String uname,password

Login

Activity diagram:



**4.2 Sequence Diagrams**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence Diagram:

Other device

System

Image decryption

Text decryption

Image encryption

User

Text encryption

Data base

1

2

3

4

5

6

7 8 9

10

11 12

1.login(fill details) 6.check data base for images 11.Decrypts the image

2.Check data base 7.Image available 12.System shows text

3.Valid/Invalid 8.encryption done

4.Choose text and encrypt 9.Send to receiver

5.Choose image 10.Receiver decrypts the text with a key

**4.3 Use Case Diagram**

A Use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Use Case Diagram:

**4.4 Collaboration Diagram**

Collaboration diagrams remains analogous with the sequence diagrams since

Diagrams encompasses various objects, links along with transmission or

receiving of Messages in this way they coordinate to structural aspects of

system.(which also provides Dynamic view of the system)

Data base

User

Image encryption

Text encryption

Application

Image decryption

Receiver

Text decryption

1

12 11 10

4 9

3

2 14

13

5 7 8

6

1. Login 8. Encrypted image
2. Check data 9. Send
3. Response 10. Response
4. Valid /invalid 11. Request key
5. Choose text 12. Decrypted key
6. Encrypted text 13. Browse the received image
7. Choose an image 14. Decrypted image

5. FUNCTIONAL DESCRIPTION

Functional requirements are those requirements that are used to illustrate the internal

working nature of the system, the description of the system and explanation of each sub

system. It consists of what task the system should perform, the processes involve, which data

should the system holds and the interfaces with the user. The functional requirements

identified areas are:

User’s registration:

The system should allow new users to register online and allow them to access.

Automatic update to database is done once new user is registered. Whenever there is new

registration the system should be able to update the database without any additional efforts

from the admin.

Non-Functional Requirements:

It describes aspects of the system that are concerned with how the system provides the

non-functional requirements i.e., it specifies the criteria that can be used to judge the

operation of a system, rather than specific behavior or functions, such as

Security:

The subsystem should provide a high level of security and integrity of the data held

by the system, only authorized personnel of the company can gain access to the company’s

secured page on the system and only users with valid password and username can login to

view user’s page.

Performance and Response time:

The system should have high performance rate when executing user’s input and

should be able to provide feedback or response within a short time span.

Error handling:

Error should be considerably minimized and an appropriate error message that guides

the user to recover from an error should be provided. Validation of user’s input is highly

essential.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 27

### What is Android

* You might have seen windows , linux and mac operating systems which are made for computers. Windows is the most popular operating system on computers. So if you know about it then it is easy for you to get an answer for what is android.
* Android is also an operating system developed by [Google](http://www.google.com). Basically it was started by some other company which was taken by Google. Google improved the operating system and made it a open source platform. It was widely adapted over the world. As it is open source it is so popular amongst the smartphones.Â  Android OS can also be used on tablet PCs.

**History of Android**

* Android, Inc. was founded in [Palo Alto, California](http://en.wikipedia.org/wiki/Palo_Alto,_California) in October 2003 by [Andy Rubin](http://en.wikipedia.org/wiki/Andy_Rubin) (co-founder of [Danger](http://en.wikipedia.org/wiki/Danger_%28company%29)), [Rich Miner](http://en.wikipedia.org/wiki/Rich_Miner) (co-founder of Wildfire Communications, Inc.), Nick Sears (once VP at [T-Mobile](http://en.wikipedia.org/wiki/T-Mobile_USA)), and Chris White (headed design and interface development at [WebTV](http://en.wikipedia.org/wiki/WebTV)) to develop, in Rubin's words "smarter mobile devices that are more aware of its owner's location and preferences".Despite the past accomplishments of the founders and early employees, Android Inc. operated secretly, revealing only that it was working on software for mobile phones. That same year, Rubin ran out of money. [Steve Perlman](http://en.wikipedia.org/wiki/Steve_Perlman), a close friend of Rubin, brought him $10,000 in cash in an envelope and refused a stake in the company.
* [Android](http://www.tech2crack.com/android/), This word means a lot in present High-Tech World. Today Smartphone are known for its [operating system](http://www.tech2crack.com/tag/operating-system/) which is Android.
* Earlier there is no option for operating systems like Android in mobile, as usual there are symbian, java featured operating systems but today things had changed a lot, everyone wants a Smartphone which is functioned on Android only.

## Android 1.0 (API level 1)

Android 1.0, the first commercial version of the software, was released on 23 September 2008. The first Android device, the [HTC Dream](http://en.wikipedia.org/wiki/HTC_Dream).

* The first Android device which runs on Android 1.0 is HTC Dream (G1), incorporated the following Android 1.0 features

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 28

*  Web Browser to show, zoom and pan full HTML and XHTML web pages, multiple pages show as windows, supporting POP3, IMAP4, and SMT
* [Android Market](http://www.tech2crack.com/tag/android-market/) app downloads and updates
* Multitasking, instant messaging, GPS, Wi-Fi an Bluetooth
* No Microsoft Exchange Server, no camcorder, youtube video player Camera Support lacked in change of camera’s resolution, quality, white balance etc
* Folder permits for the grouping of bunch of application icons into a single folder icon on the home screen

### Android 1.1 (API level 2)

On 9 February 2009, the Android 1.1 update was released, initially for the HTC Dream only. Android 1.1 was known as "[Petit Four](http://en.wikipedia.org/wiki/Petit_four)" internally, though this name was not used officially.

* This update was released for the T-Mobile G1 only. The update resolved bugs, changed the API and added a number of other features .
* Details and reviews available when a user searches for businesses on Maps.
* Longer in-call screen timeout default when using the speakerphone, plus ability to show/hide dial pad Ability to save attachments in messages.

### Android 1.5 Cupcake (API level 3)

On 30 April 2009, the Android 1.5 update was released, based on [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel) 2.6.27.

* This was the first release to officially use a codename based on a dessert item ("Cupcake"), a theme which would be used for all releases henceforth.
* The update included several new features and UI amendments
* Universal search box
* Revamped Android market Browsing categories and filters (Top free, Top paid, Just in)
* Toggle between camera and videos modes, Video recording and playback in MPEG-4 and 3GP formats.
* Faster Camera start-up and image capture, integrated photo gallery
* Much faster acquisition of GPS location (powered by SUPL AGPS)
* Auto-pairing and stereo support for Bluetooth added (A2DP and AVRCP profiles)
* On-screen soft keyboard& Directly upload videos to youtube and picassa.

.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 29



* Virtual On-screen keyboard
* Camcorder mode for recording (and watching) video
* Stereo Bluetooth
* Expanded Gesture framework and new GestureBuilder development tool
* Home screen widgets and folders
* Quick Search Box and Voice Search
* Toggle between still and video captures modes, Support for WVGA screen resolutions
* Copy/Paste and search within the browser
* Direct upload to Youtube and Picasa
* user can select multiple photos for deletion
* Battery usage indicator

### Android 2.0 Eclair (API level 5)

On 26 October 2009, the Android 2.0 SDK – codenamed Eclair – was released, based on Linux kernel 2.6.29. Changes include

* Multiple accounts for email and contact synchronization
* Microsoft Exchange Support for syncing of e-mail
* Quick Contact pop-up widget
* Search saved SMS and MMS messages
* Camera improvement include support for flash and digital zoom, scene mode, white balance, color effect and macro focus
* Keyboard improvement Adaptive Dictionary
* Bluetooth 2.1 support
* New browser User Interface and support for HTML5
* Improved Google Maps 3.1.2
* New calender features
* Besides live wallpapers, it’s allowing Home screen background images to be animated to show movement
* MotionEvent class enhanced to track multi-touch events

### Android 2.01 Eclair (API level 6)

* Minor API changes
* bug fixes and framework behavioral changes

### Android 2.1 Eclair (API level 7)

* Minor amendments to the API and bug fixes

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 30



### Android 2.2–2.2.3 Froyo (API level 8)

On 20 May 2010, the SDK for Android 2.2 (Froyo, short for [frozen yogurt](http://en.wikipedia.org/wiki/Frozen_yogurt)) was released, based on Linux kernel.

**Android 2.2 (Froyo)**

* Multiple user account
* Speed, memory, and performance optimizations
* Android market update Batch and automatic updates, installing apps to the SD card
* Improved Microsoft Exchange support
* USB tethering and Hotspot support, Voice dialing and contact sharing over Bluetooth
* Multiple keyboard languages

**Android 2.2.1**

* Bug fixes
* Security update
* Performance improvements

**Android 2.2.2**

* Fixed minor bugs
* Fixed SMS routing issues that affected the Nexus One

**Android 2.2.3**

* This update consisted of two security patches

### Android 2.3–2.3.2 Gingerbread (API level 9)

On 6 December 2010, the Android 2.3 (Gingerbread) SDK was released, based on Linux kernel

**Android 2.3 (Gingerbread)**

* Native support for more sensors
* UI refinements for simplicity and speed
* New keyboard for faster text input
* One-touch word selection and copy/paste

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 31



### Android 2.3.3–2.3.7 Gingerbread (API level 10)

**Android 2.3.3**

* Included several improvements and API fixes

**Android 2.3.4**

* Support for voice or video chat using Google Talk

**Android 2.3.5**

* Improved network performance for Nexus S 4G
* Fixed Bluetooth bug on Samsung Galaxy S
* Shadow animations for list scrolling
* Camera software enhancements
* Improved battery efficiency

**Android 2.3.6**

* Fixed a voice search bug

**Android 2.3.7**

* Introduced Google Wallet support for the Nexus S 4G

### Android 3.0 Honeycomb (API level 11)

On 22 February 2011, the Android 3.0 (Honeycomb) SDK – the first [tablet](http://en.wikipedia.org/wiki/Tablet_computer)-only Android update was released, based on Linux kernel The first device featuring this version, the [Motorola Xoom](http://en.wikipedia.org/wiki/Motorola_Xoom) tablet, was released on 24 February 2011.

* Specifically optimized for tablets and devices with larger screen sizes
* Hardware acceleration, Support for multi-core processors, Ability to encrypt all user data
* Refined multitasking, rich notifications, home screen customization, widgets, Redesigned keyboard

### Android 3.1 Honeycomb (API level 12)

* UI refinements
* Connectivity for USB accessories
* Expanded Recent Apps list
* Resizable Home screen widgets

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 32

* Support for external keyboards and pointing devices, joysticks and gamepads, FLAC audio playback
* High-performance Wi-Fi lock, HTTP proxy for each connected Wi-Fi access point

### Android 3.2 Honeycomb (API level 13)

Most first -and second-generation [Google TV](http://en.wikipedia.org/wiki/Google_TV)-enabled devices utilize Honeycomb 3.2.

**Android 3.2**

* This update first appear on Huawei’s MediaPad tablet. Changes included
* Improved hardware support, including optimizations for a wider range of tablets
* Increased ability of apps to access files on the SD card
* Compatibility display mode for apps

**Android 3.2.1**

* Bug fixes and minor security
* Stability and Wi-Fi improvements
* Update to Android Market with automatic updates and Google books
* Improved Adobe Flash support and Chinese handwriting prediction

**Android 3.2.2**

Included bug fixes and other minor improvements for the Motorola Xoom 4G.

Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)

The SDK for Android 4.0.1 (Ice Cream Sandwich), based on Linux kernel 3.0.1, was publicly released on 19 October 2011. Google's Gabe Cohen stated that Android 4.0 was "theoretically compatible" with any Android 2.3.x device in production at that time. The [source code](http://en.wikipedia.org/wiki/Source_code) for Android 4.0 became available on 14 November 2011. The update introduced numerous new features, including

**Android 4.0 (Ice-cream sandwich)**

* Increase in speed and performance
* Virtual buttons in the UI
* Separation of widgets in a new tab
* Easier-to-create folders with a drag-and-drop style
* A customizable launcher, Resizeable widgets
* Improved visual voicemail with the ability to speed up or slow down voicemail messages
* Pinch-to-zoom functionality in the calendar

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 33



* Offline search, a two-line preview, and new action bar at the bottom of the Gmail app
* Ability to swipe left or right to switch between Gmail conversations, Automatic syncing of browser with users’ Chrome bookmarks
* Integrated screenshot capture (accomplished by holding down the Power and Volume-Down buttons)
* Improved error correction on the keyboard, Real-time speech to text dictation
* Ability to access apps directly from lock screen
* Improved copy and paste functionality
* [Face Unlock](http://www.tech2crack.com/tag/face-unlock/) New feature that allows users to unlock handsets using facial identification software
* New tabbed web browser (allowing up to 16 tabs)
* Modern Roboto font
* Data Usage section in settings that lets users set warnings when they approach a certain usage limit, and disable data use when the limit is exceeded

**Android 4.0.2**

* Fixed minor bugs on the Verizon Galaxy Nexus

### Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)

**Android 4.0.3**

* Bug fixes and optimizations
* Improvements to graphics, databases, spell-checking, Bluetooth functionality
* Calendar provider enhancements, new camera apps enhancing video stabilization and QVGA resolution
* Accessibility refinements such as improved content access for screen readers

### Android 4.1 Jelly Bean (API level 16)

Google announced Android 4.1 (Jelly Bean) at the [Google I/O](http://en.wikipedia.org/wiki/Google_I/O) conference on 27 June 2012. Based on Linux kernel 3.0.31, Jelly Bean was an incremental update with the primary aim of improving the functionality and performance of the user interface. The performance improvement involved "Project Butter", which uses touch anticipation, [triple buffering](http://en.wikipedia.org/wiki/Multiple_buffering), extended [vsync](http://en.wikipedia.org/wiki/Vsync) timing and a fixed frame rate of 60 [fps](http://en.wikipedia.org/wiki/Frames_per_second) to create a fluid and "buttery-smooth" UI. Android 4.1 Jelly Bean was released to the [Android Open Source Project](http://en.wikipedia.org/wiki/AOSP#Android_Open_Source_Project) on 9 July 2012, and the [Nexus 7](http://en.wikipedia.org/wiki/Nexus_7) tablet, the first device to run Jelly Bean, was released on 13 July 2012

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 34

### Android 4.2 Jelly Bean (API level 17)

Google was expected to announce Jelly Bean 4.2 at an event in [New York City](http://en.wikipedia.org/wiki/New_York_City) on 29 October 2012, but the event was cancelled due to [Hurricane Sandy](http://en.wikipedia.org/wiki/Hurricane_Sandy). Instead of rescheduling the live event, Google announced the new version with a press release, under the slogan "A new flavor of Jelly Bean". The first devices to run Android 4.2 were [LG](http://en.wikipedia.org/wiki/LG)'s [Nexus 4](http://en.wikipedia.org/wiki/Nexus_4) and [Samsung](http://en.wikipedia.org/wiki/Samsung)'s [Nexus 10](http://en.wikipedia.org/wiki/Nexus_10), which were released on 13 November 2012.

### Usage share of Android versions

Usage share of the different versions as of February 5, 2013.

Most Android devices to date still run the older OS version 2.3 Gingerbread that was released on December 6, 2010, due to most lower-end devices still being released with it

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versions | Code Name | Release date | API level | Distribution (feb 5,2013) |
| 4.2 | [Jelly Bean](http://en.wikipedia.org/wiki/Jelly_Bean_%28operating_system%29) | November 13, 2012 | 17 | 1.4% |
| [4.1.x](http://en.wikipedia.org/wiki/Jelly_Bean_%28operating_system%29) | [Jelly Bean](http://en.wikipedia.org/wiki/Jelly_Bean_%28operating_system%29) | July 9, 2012 | 16 | 12.2% |
| 4.0.x | [Ice Cream Sandwich](http://en.wikipedia.org/wiki/Ice_Cream_Sandwich_%28operating_system%29) | December 16, 2011 | 15 | 29.0% |
| 3.2 | [Honeycomb](http://en.wikipedia.org/wiki/Honeycomb_%28operating_system%29) | July 15, 2011 | 13 | 1.0% |
| 3.1 | [Honeycomb](http://en.wikipedia.org/wiki/Honeycomb_%28operating_system%29) | May 10, 2011 | 12 | 0.3% |
| [2.3.3–2.3.7](http://en.wikipedia.org/wiki/Android_2.3) | [Gingerbread](http://en.wikipedia.org/wiki/Android_2.3) | February 9, 2011 | 10 | 45.4% |
| [2.3–2.3.2](http://en.wikipedia.org/wiki/Android_2.3) | [Gingerbread](http://en.wikipedia.org/wiki/Android_2.3) | December 6, 2010 | 9 | 0.2% |
| 2.2 | [Froyo](http://en.wikipedia.org/wiki/Android_Froyo) | May 20, 2010 | 8 | 8.1% |
| [2.0–2.1](http://en.wikipedia.org/wiki/Android_version_history#Android_2.0.2C_2.1_Eclair) | [Eclair](http://en.wikipedia.org/wiki/Android_version_history#Android_2.0.2F2.1_Eclair) | October 26, 2009 | 7 | 2.2% |
| [1.6](http://en.wikipedia.org/wiki/Donut_%28operating_system%29) | [Donut](http://en.wikipedia.org/wiki/Donut_%28operating_system%29) | September 15, 2009 | 4 | 0.2% |

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 35

### Architecture of Android

**ARCHITECTURE**  


**Fig 10.1 Architecture of Android**

**Applications**  
  
Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. Android applications are written in the Java programming language, and they run within a virtual machine (VM). It's important to note that the VM is not a JVM as we might expect, but is the Dalvik Virtual Machine, an open source

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 36





Fig 10.2 Applications

technology. Each Android application runs within an instance of the Dalvik VM, which in turn resides within a Linux-kernel managed process, as shown below. Some of basic applications include a calendar, email client, SMS program, maps, making phone calls, accessing the Web browser.

**ApplicationFramework**  
 Android is a software stack for mobile devices that includes an operating system, middleware and key applications.



Fig 10.3 Application Framework

. Underlying all applications is a set of services and systems, including

1. Activity Manager
2. Windows Manager
3. Content Provider
4. View System
5. Package Manager
6. Telephony Manager
7. Resource Manager
8. Location Manager
9. Notification Manager

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 37

Where,

* Rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.
* Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.
* Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.
* Notification Manager that enables all applications to display custom alerts in the status bar.
* An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack.

The architecture is well designed to simplify the reuse of components. Think of the application framework as a set of basictools with which adeveloper can build much more complextools.  
  
**Libraries**  
  
This layer consists of Android libraries written in C, C++, and used by various systems. These libraries tell the device how to handle different kinds of data and are exposed to Android developers via Android Application framework. Some of these libraries includes media, graphics, 3d, SQLite, web browser library etc.



Fig 10.4 Libraries

**AndroidRuntime**  
  
This layer includes set of base libraries that are required for java libraries. Every Android application gets its own instance of Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently and it executes files in executable (.Dex) optimized for minimum memory

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 38



Fig 10.5 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Dalvik Virtual Machine  
An interpreter-only virtual machine (no JIIT), register based.

* Optimized for low memory requirements
* Designed to allow multiple VM instances to run at one
* Relying on underlying OS for process isolation, memory management and threading support
* Executes Dalvik Executables (DEX) files which are zipped into an Android Package (APK)

**LinuxKernel**  
  
This layer includes Android's memory management programs, security settings, power management software and several drivers for hardware, file system access, networking and inter-process-communication. The kernel also acts as an abstraction layer between hardware and the restofthesoftwarestack.

 Fig 10.6 Linux Kernel

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 39

A Linux 2.6.24 fit for android   
  
Some common features have been removed

* No GBLIC support
* No native windowing system
* Does not include the full set of Linux utilities

New Android specific components have been added

* Alarm, Android Shared Memory
* Kernel Memory Killer, Kernel Debugger, Logger

Power Management  
Based on the standard Linux Power Management, Android has its own component.

* Application uses user space library to inform the framework about its constraints.
* Constraints are implemented using lock mechanism.

## Features

## **Application framework** enabling reuse and replacement of components

* **Dalvik virtual machine** optimized for mobile devices
* **Integrated browser** based on the open source [WebKit](http://webkit.org/) engine
* **Optimized graphics** powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
* **SQLite** for structured data storage
* **Components of Android**
* **Android Application Components**
* In this second article on Android key concepts in the [Android development tutorial](http://www.android-app-market.com/android-development-tutorial) series, we will analyze the various components by which an Android Application is made of. That is, **the**

**Android Application Components.**

* There are mainly five types of components that are used to build an application. Actually these are some objects defined in the Android SDK and provides different methods by which an application can behave.  As a developer we need only to call and extend these already defined classes to use in our application.

**These are the main Android Application Components**

* Activities
* Services
* Content Providers

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 40

* Broadcast Receivers
* Intents

I’m not going to give a full explanation on each of these components. You will learn how to use these Android application components to build applications later, when we start actual development. Now I’m giving you brief descriptions of each terms to make you familiar with the basic concepts.

## Activities

Activity is an individual user interface screen in an Android application where visual elements called Views (also known as widgets) can be placed and the user can perform various actions by interacting with it. Consider the figure. The whole window gives the user an interface to interact with and therefore this complete screen makes an Activity. The controls placed in the window allows the user to perform certain actions and are called Views or Widgets. In this example, there are five widgets, they are – TextView, EditText AnalogClock and two Buttons. The widgets in an Activity can be created in two different ways, by pure java code and by adding XML code to define the UI. The latter is always preferred. An application can have more than one Activity and each Activity operates independently, but can be linked to one another and each Activity you create must be defined in your application’s manifest file.  Each Activity in android will be subclass of Activity class defined in Android SDK.

## Services

A service is an Android application component that run in background and has no visual UI. Services are used to perform the processing parts of your application in the background. While the user is working on the foreground UI, services can be used to handle the processes that need to be done in the background. A service can be started by another Android application components such as an activity or other services and it will continue to run in the background even after the user switches to another application. Thus services are less likely to be destroyed by Android system to free resources, than Activities.

All Android services are implemented as a subclass of Service class defined in Android SDK. There are two types of services in Android.They are

**Unbound Services**

Its a type of service which is not bounded to any components. Once started, it will run in the background even after the component that started the service gets killed. It can be run in the background indefinitely and should stop by itself after the operation its intended to carry out is completed.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 41

**Bound Service**

Its bound to other components and runs only till the component to which it is bounded runs.

## Content Providers

Content providers in Android provides a flexible way to make data available across applications. Suppose you are creating any type of data in your application (For example consider you are creating a to do list in your application, then the list of things is a data) and you are storing it at any storage location, it may be in the data base, file system or in any online storage space. Then through content providers other applications are able to query, access or even modify the data you’ve created, as long as your content provider allows it. In a similar way you can access the data that other utilities have created, by using content providers. Example for content provider in Android is the contacts database. The Content provider of contacts database allows other applications to query, read, modify, and write the contacts info. Android comes with several other built in Content providers that we can use in our application. All content providers are implemented as a subclass of ContentProvider class which is defined in Android SDK.

**Broadcast Receivers**

Broadcast receivers are one of Android application components that is used to receive messages that are broadcasted by the Android system or other Android applications. There are many broadcasts that are initiated by the Android system itself and other applications can receive by using Broadcast receiver. Examples of broadcasts initiated by the system are

1. Warning that the battery is getting low

2. Screen turned off

3. Change of time zone

4. The camera has been used to take a picture

While programming, we can  use Broadcast receivers to receive these broadcasted messages and behave accordingly. Applications can also initiate broadcasts. We can initiate as many broadcasts as we want and there’s no limits for that.

## Intents

There are two types of Intents in Android

**Explicit Intents**

In explicit Intent, we are highly specific. We specify which activity should get active on receiving the intent. These are usually used for application’s internal communications.

.

ENCRYPTION AND DECRYPTION OF DATA ON IMAGES

Page 42



**Implicit Intents**

In implicit Intent we are sending a message to the Android system to find a suitable Activity that can respond to the intent. For example, to send an e-mail, we can use an intent. We will also specify  the data to be operated on, with the intent. On receiving the Intent, Android system will invoke an Activity which is able to send e-mail messages with the data that we specified. If there is more than one activity is capable of receiving the Intent, the system presents a chooser to the user so that he can select which Activity/Application should handle it.

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 43

8. CODING

ENTRY PAGE

package com.example.imageencryption;

import com.example.imageencryption.R;

import android.app.Activity;

import android.content.Intent;

import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.Menu;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

public class entry\_java extends Activity{

TextView tpwd;

Button btn\_reg,chk;

String user,password;

EditText us,pw;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.entry\_xml);

tpwd=(TextView)findViewById(R.id.textView5);

btn\_reg=(Button)findViewById(R.id.button1);

chk=(Button)findViewById(R.id.button2);

us=(EditText)findViewById(R.id.editText1);

pw=(EditText)findViewById(R.id.editText2);

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 56

tpwd.setOnClickListener(**new** OnClickListener() {

**publicvoid** onClick(View arg0) {

Intent int1=**new** Intent(getApplicationContext(),pass\_set\_java.**class**);

startActivity(int1);

}

});

btn\_reg.setOnClickListener(**new** OnClickListener() {

**publicvoid** onClick(View arg0) {

Intent int2=**new** Intent(getApplicationContext(),regster.**class**);

startActivity(int2);

}

});

chk.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View arg0) {

user=us.getText().toString();

password=pw.getText().toString();

SharedPreferences log=getSharedPreferences("regdet", ~~MODE\_WORLD\_READABLE~~);

String shruser=log.getString("username","not found");

String shrpas=log.getString("Newpassword","not found");

**if**(user.equals(shruser)&&(password.equals(shrpas)))

{

Intent chk=**new** Intent(getApplicationContext(),dec\_enc\_option\_java.**class**);

startActivity(chk);

}

**else**

{

Toast.*makeText*(getApplicationContext(), "Invalid username/password",Toast.*LENGTH\_LONG*).show();

}

}

});

}



ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 57

@Override

**publicboolean** onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.*main*, menu);

**returntrue**;

}

}

**ENC & DEC PAGE**

package com.example.imageencryption;

import com.example.imageencryption.R;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

public class dec\_enc\_option\_java extends Activity

{

Button encript\_optn\_btn,decript\_optn\_btn;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.dec\_enc\_option\_xml);

encript\_optn\_btn=(Button)findViewById(R.id.encript\_optn\_btn);

decript\_optn\_btn=(Button)findViewById(R.id.decript\_optn\_btn);

encript\_optn\_btn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent in=new Intent(getApplicationContext(),text\_encript\_java.class);

startActivity(in);

}

});

decript\_optn\_btn.setOnClickListener(new OnClickListener() {

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 58

@Override

**publicvoid** onClick(View v) {

Intent in=**new** Intent(getApplicationContext(),Imgdecript.**class**);

startActivity(in);

}

});

}

}

Text Encryption PAGE

**package** com.example.imageencryption;

**import** com.example.imageencryption.R;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.Toast;

**publicclass** text\_encript\_java **extends** Activity{

EditText enc\_text\_key\_edt,enc\_text\_edt;

Button enc\_text\_btn;

**publicstatic** String *encrypted\_string*;

SimpleCrypto2 sicrypto=**new** SimpleCrypto2();

@Override

**protectedvoid** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*text\_encript\_xml*);

enc\_text\_key\_edt=(EditText)findViewById(R.id.*enc\_text\_key\_edt*);

enc\_text\_edt=(EditText)findViewById(R.id.*enc\_text\_edt*);

enc\_text\_btn=(Button)findViewById(R.id.*enc\_text\_btn*);

enc\_text\_btn.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View v) {

**try** {

//text.setText(SimpleCrypto.encrypt("ganesh", "Hai how r u?"));

*encrypted\_string*=SimpleCrypto2.*encrypt*(enc\_text\_key\_edt.getText().toString(),enc\_text\_edt.getText().toString());

//text.setText(encrypted\_string);

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 59

Toast.*makeText*(getApplicationContext(),*encrypted\_string* , 100).show();

enc\_text\_key\_edt.setText("");

enc\_text\_edt.setText("");

Intent encriptImg\_intent=**new** Intent(getApplicationContext(),Imgencript.**class**);

startActivity(encriptImg\_intent);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

}

IMAGE ENCRYPTION

**package** com.example.imageencryption;

**import** java.io.File;

**import** java.io.FileOutputStream;

**import** java.io.OutputStream;

**import** com.example.imageencryption.R;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.database.Cursor;

**import** android.graphics.Bitmap;

**import** android.graphics.BitmapFactory;

**import** android.net.Uri;

**import** android.os.Bundle;

**import** android.os.Environment;

**import** android.provider.MediaStore;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.ImageView;

**import** android.widget.TextView;

**import** android.widget.Toast;

**publicclass** Imgencript **extends** Activity{

Button encript\_browse\_btn,enc\_savecode\_btn,enc\_sendimg\_btn;

EditText enc\_txtcode\_edt;

TextView enc\_img\_name;

ImageView enc\_brows\_image;

Bitmap bmpImage,bmpImage2;

@Override

**protectedvoid** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*imgencript\_xml*);

encript\_browse\_btn=(Button)findViewById(R.id.*encript\_browse\_btn*);

enc\_savecode\_btn=(Button)findViewById(R.id.*enc\_savecode\_btn*);

enc\_sendimg\_btn=(Button)findViewById(R.id.*enc\_sendimg\_btn*);

enc\_txtcode\_edt=(EditText)findViewById(R.id.*enc\_txtcode\_edt*);

enc\_img\_name=(TextView)findViewById(R.id.*enc\_img\_name*);

enc\_brows\_image=(ImageView)findViewById(R.id.*enc\_brows\_image*);

enc\_txtcode\_edt.setText(text\_encript\_java.*encrypted\_string*);

encript\_browse\_btn.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View v) {

Intent photoPickerIntent = **new** Intent(Intent.*ACTION\_PICK*);

photoPickerIntent.setType("image/\*");

startActivityForResult(photoPickerIntent, 1337);

}

});

enc\_savecode\_btn.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View v) {

//do call encription here............

**if**(bmpImage == **null**) {

Toast.*makeText*(getBaseContext(), "No image selected, cannot encode",

Toast.*LENGTH\_SHORT*).show();

**return**;

}

**if**(enc\_txtcode\_edt.getText().length() < 1) {

Toast.*makeText*(getBaseContext(), "No secret message, cannot encode",

Toast.*LENGTH\_SHORT*).show();

**return**;

}

bmpImage = Steganography.*encode*(bmpImage, enc\_txtcode\_edt.getText().toString());

//String path = Environment.getExternalStorageDirectory().toString();

OutputStream fOut = **null**;

File file=**new** File(Environment.*getExternalStorageDirectory*(),"a"+enc\_img\_name.getTex() .toString());

**try**{

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

Page 60

fOut = **new** FileOutputStream(file);

bmpImage.compress(Bitmap.CompressFormat.*PNG*, 100, fOut);

fOut.flush();

fOut.close();

enc\_txtcode\_edt.setText("");

//MediaStore.Images.Media.insertImage(getContentResolver(),bmpImage2, "a"+enc\_img\_name.getText().toString(), null);

//MediaStore.Images.Media.insertImage(getContentResolver(), Environment.getExternalStorageDirectory()+"/a"+enc\_img\_name.getText().toString(), "abc", null);

Toast.*makeText*(getBaseContext(), "Secret encoded successfully and image saved",

Toast.*LENGTH\_SHORT*).show();

}**catch**(Exception ex){

System.*out*.println(ex.getMessage());

}

}

});

enc\_sendimg\_btn.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View v) {

//do sending file here............

Intent intent = **new** Intent();

intent.setAction(Intent.*ACTION\_SEND*);

intent.setType("image/\*");

String uri = "/mnt/sdcard/test.jpg";

intent.putExtra(Intent.*EXTRA\_STREAM*, Uri.*fromFile*(**new** File(Environment.*getExternalStorageDirectory*(), "a"+enc\_img\_name.getText().toString())));

startActivity(intent);

}

});

}

// image picker intent

@Override

**publicvoid** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {

**super**.onActivityResult(requestCode, resultCode, data);

**if**(requestCode == 1337) {

**if**(resultCode == Activity.*RESULT\_OK*){

Uri selectedImage = data.getData();

enc\_brows\_image.setImageURI(selectedImage);

String sFilePath = getRealPathFromURI(selectedImage);

**if**(sFilePath == **null**) {

Toast.*makeText*(getBaseContext(), "Image not found",

Toast.*LENGTH\_SHORT*).show();

**return**;

} **else** {

String saPathParts[] = sFilePath.split("/");

String sFileName = saPathParts[saPathParts.length-1];

enc\_img\_name.setText(sFileName);

bmpImage = BitmapFactory.*decodeFile*(sFilePath);

**if**(bmpImage == **null**) {

Toast.*makeText*(getBaseContext(), "Image invalid",

Toast.*LENGTH\_SHORT*).show();

**return**;

}

}

}

}

}

// gets the actual file path from a Uri object

**private** String getRealPathFromURI(Uri contentURI) {

Cursor cursor = getContentResolver().query(contentURI, **null**, **null**, **null**, **null**);

cursor.moveToFirst();

**int** idx = cursor.getColumnIndex(MediaStore.Images.ImageColumns.*DATA*);

**return** cursor.getString(idx);

}

}

**TEXT DECRYPTION**

package com.example.imageencryption;

import com.example.imageencryption.R;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

public class text\_decript\_java extends Activity{

EditText dec\_text\_key\_edt;

Button dec\_text\_btn;

TextView dec\_text\_view;

int count=0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.text\_decript\_xml);

dec\_text\_btn=(Button)findViewById(R.id.dec\_text\_btn);

dec\_text\_key\_edt=(EditText)findViewById(R.id.dec\_text\_key\_edt);

dec\_text\_view=(TextView)findViewById(R.id.dec\_text\_view);

dec\_text\_btn.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View v) {

Toast.*makeText*(getApplicationContext(), "code to be writen"+Imgdecript.*text\_code\_string*, 100).show();

**try** {

//System.out.println("Decrypted ----: "+SimpleCrypto.decrypt("ganesh", SimpleCrypto.encrypt("ganesh", "Hai how r u?")));

dec\_text\_view.setText(SimpleCrypto2.*decrypt*(dec\_text\_key\_edt.getText().toString(), Imgdecript.*text\_code\_string*));

} **catch** (Exception e) {

e.printStackTrace();

Toast.*makeText*(getApplicationContext(), "criptography exception see log cat....", 100).show();

count++;

**if**(count==3){

Intent in=**new** Intent(getApplicationContext(),entry\_java.**class**);

startActivity(in);

finish();

}

}

}

});

}

}

**IMAGE DECRYPTION**

package com.example.imageencryption;

import com.example.imageencryption.R;

import android.app.Activity;

import android.content.Intent;

import android.database.Cursor;

import android.graphics.Bitmap;

import android.graphics.BitmapFactory;

`

import android.net.Uri;

import android.os.Bundle;

import android.provider.MediaStore;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

public class Imgdecript extends Activity{

Button decript\_browse\_btn,dec\_getcode\_btn,dec\_gettext\_btn;

EditText dec\_txtcode\_edt;

TextView dec\_img\_name;

ImageView dec\_brows\_image;

Bitmap bmpImage;

Public static String text\_code\_string;

@Override

protected void onCreate(Bundle savedInstanceState)

{

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

decript\_browse\_btn=(Button)findViewById(R.id.decript\_browse\_btn);

dec\_getcode\_btn=(Button)findViewById(R.id.dec\_getcode\_btn);

dec\_gettext\_btn=(Button)findViewById(R.id.dec\_gettext\_btn);

dec\_txtcode\_edt=(EditText)findViewById(R.id.dec\_txtcode\_edt);

dec\_img\_name=(TextView)findViewById(R.id.dec\_img\_name);

dec\_brows\_image=(ImageView)findViewById(R.id.dec\_brows\_image);

decript\_browse\_btn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

Intent photoPickerIntent = new Intent(Intent.ACTION\_PICK);

photoPickerIntent.setType("image/\*");

startActivityForResult(photoPickerIntent, 1337);

}

});

dec\_getcode\_btn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

if(bmpImage == null) {

Toast.makeText(getBaseContext(), "No image selected, cannot decode",

Toast.LENGTH\_SHORT).show();

return;

}

String decodedMsg = Steganography.decode(bmpImage);

text\_code\_string=decodedMsg;

if(decodedMsg == null || decodedMsg.length() < 1) {

Toast.makeText(getBaseContext(), "There was no message encoded in the image",

Toast.LENGTH\_SHORT).show();

return;

} else {

dec\_txtcode\_edt.setText(decodedMsg);

Toast.makeText(getBaseContext(), "Secret decrypted from image!",

Toast.LENGTH\_SHORT).show();

}

}

});

dec\_gettext\_btn.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

//navigate to text decription page here............

Intent text\_dec\_intent=new Intent(getApplicationContext(),text\_decript\_java.class);

startActivity(text\_dec\_intent);

}

});

}

// image picker intent

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if(requestCode == 1337) {

if(resultCode == Activity.RESULT\_OK){

Uri selectedImage = data.getData();

dec\_brows\_image.setImageURI(selectedImage);

String sFilePath = getRealPathFromURI(selectedImage);

if(sFilePath == null) {

Toast.makeText(getBaseContext(), "Image not found",

Toast.LENGTH\_SHORT).show();

return;

} else {

String saPathParts[] = sFilePath.split("/");

String sFileName = saPathParts[saPathParts.length-1];

dec\_img\_name.setText(sFileName);

bmpImage = BitmapFactory.decodeFile(sFilePath);

if(bmpImage == null) {

Toast.makeText(getBaseContext(), "Image invalid",

Toast.LENGTH\_SHORT).show();

return;

}

}

}

}

}

// gets the actual file path from a Uri object

private String getRealPathFromURI(Uri contentURI) {

Cursor cursor = getContentResolver().query(contentURI, null, null, null, null);

cursor.moveToFirst();

int idx = cursor.getColumnIndex(MediaStore.Images.ImageColumns.DATA);

return cursor.getString(idx);

}

}

**REGISTER PAGE**

**package** com.example.imageencryption;

**import** com.example.imageencryption.R;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.content.SharedPreferences;

**import** android.content.SharedPreferences.Editor;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.Toast;

**publicclass** regster **extends** Activity{

EditText ustxt;

EditText nptxt, cnptxt, sqtxt, anstxt;

Button reg;

String user,paswrd,cnpaswrd,seq,ans;

**protectedvoid** onCreate(Bundle savedInstanceState) {

// **TODO** Auto-generated method stub

**super**.onCreate(savedInstanceState);

setContentView(R.layout.register);

reg=(Button)findViewById(R.id.button1);

ustxt=(EditText)findViewById(R.id.editText1);

nptxt=(EditText)findViewById(R.id.editText2);

cnptxt=(EditText)findViewById(R.id.editText3);

sqtxt=(EditText)findViewById(R.id.editText4);

anstxt=(EditText)findViewById(R.id.editText5);

reg.setOnClickListener(**new** OnClickListener() {

@Override

**publicvoid** onClick(View arg0) {

user=ustxt.getText().toString();

paswrd=nptxt.getText().toString();

cnpaswrd=cnptxt.getText().toString();

seq=sqtxt.getText().toString();

ans=anstxt.getText().toString();

**if**(paswrd.equals(cnpaswrd))

{

SharedPreferences spf=getSharedPreferences("regdet",MODE\_WORLD\_WRITEABLE);

Editor regis=spf.edit();

regis.putString("username", user);

regis.putString("Newpassword", paswrd);

regis.putString("Sequrity", seq);

regis.putString("Answer", ans);

regis.commit();

Intent in=**new** Intent(getApplicationContext(),entry\_java.**class**);

startActivity(in);

}

**else**

{

Toast.makeText(getApplicationContext(),"password and confirm password didnot match" ,Toast.LENGTH\_LONG).show();

}

}

});

}

}

PASSWORD SETTINGS

package com.example.imageencryption;

import com.example.imageencryption.R;

import android.app.Activity;

import android.content.SharedPreferences;

import android.content.SharedPreferences.Editor;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

public class pass\_set\_java extends Activity{

Button save,ok;

EditText oldpsd,newpsd,cnpsd,sq,sqans;

String old,chng,cnchng,seq,seqans;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.pass\_set\_xml);

save=(Button)findViewById(R.id.button2);

ok=(Button)findViewById(R.id.button1);

oldpsd=(EditText)findViewById(R.id.editText1);

newpsd=(EditText)findViewById(R.id.editText2);

cnpsd=(EditText)findViewById(R.id.editText3);

sq=(EditText)findViewById(R.id.editText4);

sqans=(EditText)findViewById(R.id.editText5);

save.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

old=oldpsd.getText().toString();

chng=newpsd.getText().toString();

cnchng=cnpsd.getText().toString();

if(chng.equals(cnchng))

{

SharedPreferences pass=getSharedPreferences("regdet", MODE\_WORLD\_WRITEABLE);

Editor psd=pass.edit();

psd.putString("Newpassword", chng);

psd.commit();

}

else

{

Toast.makeText(getApplicationContext(), "Password and Confirm Password donot match", Toast.LENGTH\_LONG).show();

}

}

});

ok.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

seq=sq.getText().toString();

seqans=sqans.getText().toString();

SharedPreferences forget=getSharedPreferences("regdet", MODE\_WORLD\_WRITEABLE);

String shrseq=forget.getString("Sequrity", "Not found");

String shrans=forget.getString("Answer", "Not found");

if(seq.equals(shrseq)&&(seqans.equals(shrans)))

{

String shrpas=forget.getString("Newpassword", "Not found");

Toast.makeText(getApplicationContext(), "Your Password is:"+shrpas, Toast.LENGTH\_LONG).show();

}

else

{

Toast.makeText(getBaseContext(), "Please Enter correct Security Question and Answer", Toast.LENGTH\_LONG).show();

}

}

});

}

}

LSB2BIT PAGE

package com.example.imageencryption;

/\*\*

\* Copyright (C) 2009 Pasquale Paola

This library is free software; you can redistribute it and/or

modify it under the terms of the GNU Lesser General Public

License as published by the Free Software Foundation; either

version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU

Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public

License along with this library; if not, write to the Free Software

Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

\*/

import java.util.Vector;

import android.util.Log;

/\*\*

\* Core algorithm of MobiStego.

\* @author <a href="mailto:pasquale.paola@gmail.com">Pasquale Paola</a>

\*

\*/

public class LSB2bit {

private static int[] binary = { 16, 8, 0 };

private static byte[] andByte = { (byte) 0xC0, 0x30, 0x0C, 0x03 };

private static int[] toShift = { 6, 4, 2, 0 };

public static String END\_MESSAGE\_COSTANT = "#!@";

public static String START\_MESSAGE\_COSTANT = "@!#";

/\*\*

\* This method represent the core of LSB on 2 bit (Encoding).

\* @param oneDPix The <b>rgb</b> array.

\* @param imgCols Image width.

\* @param imgRows Image height.

\* @param str Message to encode.

\* @param hand A handler interface, for the progress bar.

\* @return Encoded message image.

\*/

public static byte[] encodeMessage(int[] oneDPix, int imgCols, int imgRows,

String str) {

str += END\_MESSAGE\_COSTANT;

str = START\_MESSAGE\_COSTANT + str;

byte[] msg = str.getBytes();

int channels = 3;

int shiftIndex = 4;

//Array.newInstance(Byte.class, imgRows \* imgCols \* channels);

byte[] result = new byte[imgRows \* imgCols \* channels];

int msgIndex = 0;

int resultIndex = 0;

boolean msgEnded = false;

for (int row = 0; row < imgRows; row++) {

for (int col = 0; col < imgCols; col++) {

int element = row \* imgCols + col;

byte tmp = 0;

for (int channelIndex = 0; channelIndex < channels; channelIndex++) {

if (!msgEnded) {

tmp = (byte) ((((oneDPix[element] >> binary[channelIndex]) & 0xFF) & 0xFC) | ((msg[msgIndex] >> toShift[(shiftIndex++)

% toShift.length]) & 0x3));// 6

if (shiftIndex % toShift.length == 0) {

msgIndex++;

}

if (msgIndex == msg.length) {

msgEnded = true;

}

} else {

tmp = (byte) ((((oneDPix[element] >> binary[channelIndex]) & 0xFF)));

}

result[resultIndex++] = tmp;

}

}

}

return result;

}

/\*\*

\* This is the decoding method of LSB on 2 bit.

\* @param oneDPix The byte array image.

\* @param imgCols Image width.

\* @param imgRows Image height.

\* @return The decoded message.

\*/

public static String decodeMessage(byte[] oneDPix, int imgCols,

int imgRows) {

Vector<Byte> v = new Vector<Byte>();

String builder = "";

int shiftIndex = 4;

byte tmp = 0x00;

for (int i = 0; i < oneDPix.length; i++) {

tmp = (byte) (tmp | ((oneDPix[i] << toShift[shiftIndex

% toShift.length]) & andByte[shiftIndex++ % toShift.length]));

if (shiftIndex % toShift.length == 0) {

v.addElement(new Byte(tmp));

byte[] nonso = { (v.elementAt(v.size() - 1)).byteValue() };

String str = new String(nonso);

// if (END\_MESSAGE\_COSTANT.equals(str)) {

if (builder.endsWith(END\_MESSAGE\_COSTANT)) {

break;

} else {

builder = builder + str;

if (builder.length() == START\_MESSAGE\_COSTANT.length()

&& !START\_MESSAGE\_COSTANT.equals(builder)) {

builder = null;

break;

}

}

tmp = 0x00;

}

}

if (builder != null)

builder = builder.substring(START\_MESSAGE\_COSTANT.length(), builder

.length()

- END\_MESSAGE\_COSTANT.length());

return builder;

}

/\*\*

\* Convert the byte array to an int array.

\* @param b The byte array.

\* @return The int array.

\*/

public static int[] byteArrayToIntArray(byte[] b) {

Log.v("Size byte array", b.length+"");

int size=b.length / 3;

Log.v("Size Int array",size+"");

System.runFinalization();

System.gc();

Log.v("FreeMemory", Runtime.getRuntime().freeMemory()+"");

int[] result = new int[size];

int off = 0;

int index = 0;

while (off < b.length) {

result[index++] = byteArrayToInt(b, off);

off = off + 3;

}

return result;

}

/\*\*

\* Convert the byte array to an int.

\*

\* @param b

\* The byte array

\* @return The integer

\*/

public static int byteArrayToInt(byte[] b) {

return byteArrayToInt(b, 0);

}

/\*\*

\* Convert the byte array to an int starting from the given offset.

\* @param b

\* The byte array

\* @param offset

\* The array offset

\* @return The integer

\*/

public static int byteArrayToInt(byte[] b, int offset) {

int value = 0x00000000;

for (int i = 0; i < 3; i++) {

int shift = (3 - 1 - i) \* 8;

value |= (b[i + offset] & 0x000000FF) << shift;

}

value = value & 0x00FFFFFF;

return value;

}

/\*\*

\* Convert integer array representing [argb] values to byte array

\* representing [rgb] values

\*

\* @param array Integer array representing [argb] values.

\* @return byte Array representing [rgb] values.

\*/

public static byte[] convertArray(int[] array) {

byte[] newarray = new byte[array.length \* 3];

for (int i = 0; i < array.length; i++) {

/\*

\* newarray[i \* 3] = (byte) ((array[i]) & 0xFF); newarray[i \* 3 + 1]

\* = (byte)((array[i] >> 8)& 0xFF); newarray[i \* 3 + 2] =

\* (byte)((array[i] >> 16)& 0xFF);

\*/

newarray[i \* 3] = (byte) ((array[i] >> 16) & 0xFF);

newarray[i \* 3 + 1] = (byte) ((array[i] >> 8) & 0xFF);

newarray[i \* 3 + 2] = (byte) ((array[i]) & 0xFF);

}

return newarray;

}

}

STENOGRAPHY

package com.example.imageencryption;

//import java.io.ByteArrayOutputStream;

//import java.nio.ByteBuffer;

//import java.nio.IntBuffer;

import android.graphics.Bitmap;

import android.graphics.Bitmap.Config;

//import android.graphics.BitmapFactory;

import android.graphics.Color;

/

\* @file Steganography.java

\* @author Ian Lamb

\* @author Derek Brown

\* @version 2.0

\* @created Dec 04, 2012

\*/

public class Steganography

{

public Steganography() {}

// encodes secret to bitmap

public static Bitmap encode(Bitmap bmp, String secret)

{

int height = bmp.getHeight();

int width = bmp.getWidth();

Bitmap newImage = null;

int[] imgPixels = new int[width \* height];

bmp.getPixels(imgPixels, 0, width, 0, 0, width, height);

int density = bmp.getDensity();

bmp.recycle();

try

{

byte[] byteImage = LSB2bit.encodeMessage(imgPixels, width, height, secret);

newImage = Bitmap.createBitmap(width, height, Config.ARGB\_8888);

newImage.setDensity(density);

int imgMod[] = LSB2bit.byteArrayToIntArray(byteImage);

int masterIndex = 0;

for (int j = 0; j < height; j++)

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

// The unique way to write correctly the sourceBitmap, android bug!!!

newImage.setPixel(i, j, Color.argb(0xFF,

imgMod[masterIndex] >> 16 & 0xFF,

imgMod[masterIndex] >> 8 & 0xFF,

imgMod[masterIndex++] & 0xFF));

}

}

catch(Exception e)

{

System.out.println(e.getMessage());

}

return newImage;

}

// decodes secret from bitmap

public static String decode(Bitmap bmp)

{

byte[] b = null;

try

{

int[] pixels = new int[bmp.getWidth() \* bmp.getHeight()];

bmp.getPixels(pixels, 0, bmp.getWidth(), 0, 0, bmp.getWidth(), bmp.getHeight());

b = LSB2bit.convertArray(pixels);

}

catch (OutOfMemoryError er)

{

System.out.println( "Image too large, out of memory!");

}

final String vvv = LSB2bit.decodeMessage(b, bmp.getWidth(), bmp.getHeight());

return vvv;

}

/\*\* We tried doing our own bit manipulation but we ran into problems,

\* so we ended up using an open source library called LSB2bit

\* We left our attempted code to show our effort! :D

// get byte data from bitmap

private static byte[] getByteData(Bitmap bmp)

{

// ByteArrayOutputStream stream = new ByteArrayOutputStream();

// bmp.compress(Bitmap.CompressFormat.PNG, 100, stream);

// byte[] bytes = stream.toByteArray();

int nBytes = bmp.getWidth()\*bmp.getHeight()\*4; //calculate how many bytes our image consists of. Use a different value than 4 if you don't use 32bit images.

ByteBuffer buffer = ByteBuffer.allocate(nBytes); //Create a new buffer

bmp.copyPixelsToBuffer(buffer); //Move the byte data to the buffer

byte[] bytes = buffer.array(); //Get the underlying array containing the data.

return bytes;

}

private static ByteBuffer byteArrayToByteBuffer( byte[] bytes )

{

ByteBuffer buffer = ByteBuffer.allocate( bytes.length );

buffer.put( bytes );

buffer.position(0);

return buffer;

}

private static IntBuffer byteArrayToIntBuffer( byte[] bytes )

{

ByteBuffer buffer = ByteBuffer.allocate( bytes.length );

buffer.put( bytes );

//buffer.position(0);

return buffer.asIntBuffer();

}

// // get bitmap from byte data

// private static Bitmap bytesToBitmap(byte[] bytes, int width, int height )

// {

//// Bitmap bmp = null;

//// try{

//// bmp = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);

//// }catch(Exception ex){

//// System.out.println(ex.getMessage());

//// }

//// return bmp;

//

// int nBytes = height\*width\*4; //calculate how many bytes our image consists of. Use a different value than 4 if you don't use 32bit images.

// ByteBuffer buffer = ByteBuffer.allocate(nBytes);

// buffer.put( bytes );

// buffer.position(0);

//

// Bitmap bmp = null;

// try

// {

// //bmp.copyPixelsFromBuffer( buffer );

// }

// catch( Exception ex )

// {

// System.out.println( ex.getMessage() );

// }

//

// return bmp;

// }

// convert int to byte array

private static byte[] bitConversion(int i)

{

//originally integers (ints) cast into bytes

//byte byte7 = (byte)((i & 0xFF00000000000000L) >>> 56);

//byte byte6 = (byte)((i & 0x00FF000000000000L) >>> 48);

//byte byte5 = (byte)((i & 0x0000FF0000000000L) >>> 40);

//byte byte4 = (byte)((i & 0x000000FF00000000L) >>> 32);

//only using 4 bytes

byte byte3 = (byte)((i & 0xFF000000) >>> 24); //0

byte byte2 = (byte)((i & 0x00FF0000) >>> 16); //0

byte byte1 = (byte)((i & 0x0000FF00) >>> 8 ); //0

byte byte0 = (byte)((i & 0x000000FF) );

//{0,0,0,byte0} is equivalent, since all shifts >=8 will be 0

return(new byte[]{byte3,byte2,byte1,byte0});

}

// encodes the text

private static byte[] encodeText(byte[] image, byte[] addition, int offset)

{

//check that the data + offset will fit in the image

if(addition.length + offset > image.length)

{

throw new IllegalArgumentException("File not long enough!");

}

//loop through each addition byte

for(int i=0; i<addition.length; ++i)

{

//loop through the 8 bits of each byte

int add = addition[i];

for(int bit=7; bit>=0; --bit, ++offset) //ensure the new offset value carries on through both loops

{

//assign an integer to b, shifted by bit spaces AND 1

//a single bit of the current byte

int b = (add >>> bit) & 1;

//assign the bit by taking: [(previous byte value) AND 0xfe] OR bit to add

//changes the last bit of the byte in the image to be the bit of addition

image[offset] = (byte)((image[offset] & 0xFE) | b );

}

}

return image;

}

// decodes the text

private static byte[] decodeText(byte[] image)

{

int length = 0;

int offset = 32;

//loop through 32 bytes of data to determine text length

for(int i=0; i<32; ++i) //i=24 will also work, as only the 4th byte contains real data

{

length = (length << 1) | (image[i] & 1);

}

byte[] result = null;

try

{

result = new byte[length];

}

catch( Exception ex )

{

System.out.println("Error, the size of the message to be read was too large, causing: " + e x.getMessage() );

}

//loop through each byte of text

for(int b=0; b<result.length; ++b )

{

//loop through each bit within a byte of text

for(int i=0; i<8; ++i, ++offset)

{

//assign bit: [(new byte value) << 1] OR [(text byte) AND 1]

result[b] = (byte)((result[b] << 1) | (image[offset] & 1));

}

}

return result;

}

\*/

}

CRYPTOGRAPHY

package com.example.imageencryption;

import java.security.SecureRandom;

import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;

public class SimpleCrypto2 {

public static String encrypt(String seed, String cleartext) throws Exception {

byte[] rawKey = getRawKey(seed.getBytes());

byte[] result = encrypt(rawKey, cleartext.getBytes());

return toHex(result);

}

public static String decrypt(String seed, String encrypted) throws Exception {

byte[] rawKey = getRawKey(seed.getBytes());

byte[] enc = toByte(encrypted);

byte[] result = decrypt(rawKey, enc);

return new String(result);

}

private static byte[] getRawKey(byte[] seed) throws Exception {

KeyGenerator kgen = KeyGenerator.getInstance("AES");

SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");

sr.setSeed(seed);

kgen.init(128, sr); // 192 and 256 bits may not be available

SecretKey skey = kgen.generateKey();

byte[] raw = skey.getEncoded();

return raw;

}

private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.ENCRYPT\_MODE, skeySpec);

byte[] encrypted = cipher.doFinal(clear);

return encrypted;

}

private static byte[] decrypt(byte[] raw, byte[] encrypted) throws Exception {

SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

Cipher cipher = Cipher.getInstance("AES");

cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

byte[] decrypted = cipher.doFinal(encrypted);

return decrypted;

}

public static String toHex(String txt) {

return toHex(txt.getBytes());

}

public static String fromHex(String hex) {

return new String(toByte(hex));

}

public static byte[] toByte(String hexString) {

int len = hexString.length()/2;

byte[] result = new byte[len];

for (int i = 0; i < len; i++)

result[i] = Integer.valueOf(hexString.substring(2\*i, 2\*i+2), 16).byteValue();

return result;

}

public static String toHex(byte[] buf) {

if (buf == null)

return "";

StringBuffer result = new StringBuffer(2\*buf.length);

for (int i = 0; i < buf.length; i++) {

appendHex(result, buf[i]);

}

return result.toString();

}

private final static String HEX = "0123456789ABCDEF";

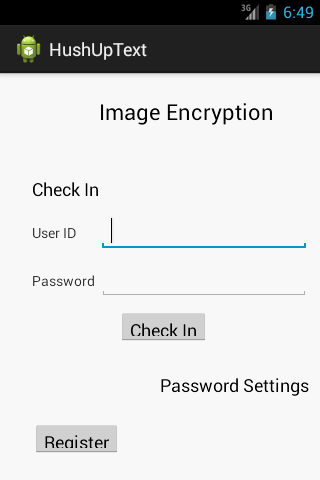
private static void appendHex(StringBuffer sb, byte b) {

sb.append(HEX.charAt((b>>4)&0x0f)).append(HEX.charAt(b&0x0f));

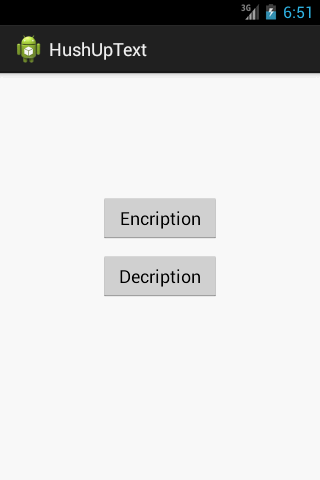
}

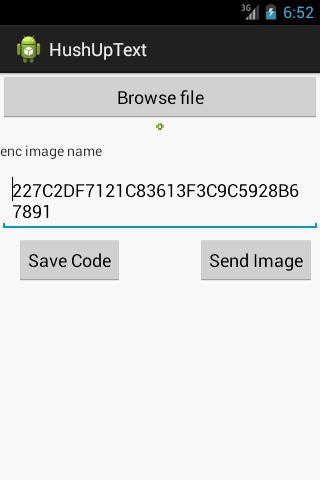
}

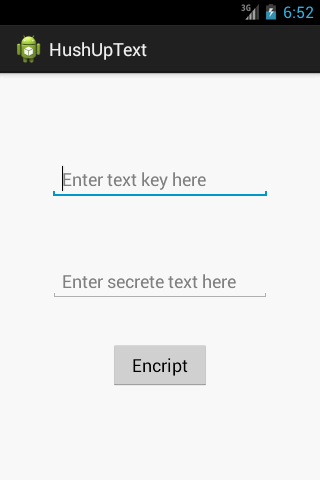
OUTPUT SCREENS











1. SYSTEM TESTING AND IMPLEMENTATION

**TESTING**

Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model.Other test levels are classified by the testing objective.

**5.1Unit Testing**

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructor.]

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

**5.2 Integration Testing**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be localised more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

**5.3 System Testing**

System testing tests a completely integrated system to verify that it meets its requirements.

Acceptance Testing

At last the system is delivered to the user for Acceptance testing

Page 61

ENHANCED DATA SECURITY MODEL IN CLOUD COMPUTING

 **BIBILOGRAPHY**

**Text books Referred**

Software Engineering: Pressman

Database System concepts: Korth

Unified Modeling Language: Grady Booch

System Analysis and Design: James Senn

## Programming Books

1. Hello, Android, E. Burnette, The Pragmatic Programmers (2009).
2. Professional Android 2 Application Development, R. Meier, Wiley (2010).
3. Beginning Android 2, M. Murphy, Apress (2010).
4. Android Wireless Application Development, S. Conder and L. Darcey, Addison-Wesley (2010).
5. Android Application Development in 24 Hours , L. Darcey and S. Conder, Sams (2010).
6. The Android Developer's Cookbook, J. Steele, N. To, Addison-Wesley (2011).

## Programmer Web Resources

1. Android Developer Guide: <http://developer.android.com/guide/index.html>.
2. Android API: <http://developer.android.com/reference/packages.html>

Page 62

1. Java 6 API: <http://download-llnw.oracle.com/javase/6/docs/api/>
2. Google Maps API: <http://code.google.com/android/add-ons/google-apis/reference/com/google/android/maps/package-summary.html>
3. Android Fundamentals: <http://developer.android.com/guide/topics/fundamentals.html>
4. The Java Tutorials: <http://download-llnw.oracle.com/javase/tutorial/index.html>
5. Android Native Development Kit: <http://developer.android.com/sdk/ndk/index.html>
6. Android User Interfaces: <http://developer.android.com/guide/topics/ui/index.html>
7. Declaring Layout: <http://developer.android.com/guide/topics/ui/declaring-layout.html>
8. Common Tasks: <http://developer.android.com/guide/appendix/faq/commontasks.html>
9. Maps External Library:

<http://code.google.com/android/add-ons/google-apis/maps-overview.html>

1. Maps API Key: <http://code.google.com/android/add-ons/google-apis/mapkey.html>
2. Icons: <http://developer.android.com/guide/practices/ui_guidelines/icon_design.html>
3. Sample Source Code: <http://developer.android.com/resources/samples/get.html>
4. List of Sample Apps: <http://developer.android.com/resources/samples/index.html>.
5. apps-for-android Sample Apps: <http://code.google.com/p/apps-for-android/>.
6. Android Developer's Blog:<http://android-developers.blogspot.com/>
7. Developer FAQ: <http://developer.android.com/resources/faq/>
8. Developer Forums: <http://developer.android.com/resources/community-groups.html>
9. Android Developer's Group: <http://groups.google.com/group/android-developers?lnk=>
10. XDA-Developers Forums: <http://forum.xda-developers.com/>

## General Android Information and/or Gossip

1. Android Power: <http://blogs.computerworld.com/raphael>
2. The Droid Guy: <http://thedroidguy.com/>
3. Seeing through Windows: <http://blogs.computerworld.com/gralla>
4. Phandroid: <http://phandroid.com/>
5. Android Guys: <http://www.androidguys.com/>
6. CW: <http://www.computerworld.com/s/article/9178688/Android_news_reviews_more>
7. Wireless & Mobile News: <http://www.wirelessandmobilenews.com/tag/android>