GULALA ALI HAMA AMIN SHARING SERVICES USING ANDROID INTERFACE DEFINITION LANGUAGE (AIDL) NEU 2017

SHARING SERVICES USING ANDROID INTERFACE DEFINITION LANGUAGE (AIDL)

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES OF NEAR EAST UNIVERSITY

By

GULALA ALI HAMA AMIN

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Systems Engineering

NICOSIA, 2017

SHARING SERVICES USING ANDROID INTERFACE DEFINITION LANGUAGE

(AIDL)

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF APPLIED SCIENCES

OF

NEAR EAST UNIVERSITY

By

GULALA ALI HAMA AMIN

In Partial Fulfillment of the Requirements for

the Degree of Master of Science

in

Information Systems Engineering

NICOSIA, 2017

GULALA ALI HAMA AMIN: SHARING SERVICES USING ANDROID INTERFACE DEFINITION LANGUAGE (AIDL)

Approval of Director of Graduate School of Applied Sciences

Prof. Dr. Nadire ÇAVUŞ

We certify this thesis is satisfactory for the award of the degree of Masters of Science in Information Systems Engineering

Examining Committee in Charge:

Assist.Prof.Dr. Yöney Kırsal Ever	Department of Software Engineering, NEU
Assist.Prof.Dr. Hüseyin Lort	Department of Computer and Instructional Technology Teaching, GAU
Assist.Prof.Dr. Boran Şekeroğlu	Department of Information Systems Engineering, NEU

I hereby declare that all information in this dissertation has been obtained and presented in accordance with academic rules and ethical conducts. I also declare that, as required by these rules and conducts, I have fully cited and referenced all material and results that are not original to this work.

Gulala Hama Amin

Signature:

Date:

ACKNOWLEDGEMENTS

I would first like to thank God for giving me the desire and determination to learn and to love scientific arena.

I would also like to thank my supervisor Assist. Prof. Dr. Boran ŞEKEROĞLU for his continuous support, his excellent cooperation, whenever I had a question about my research or writing, he consistently allowed this thesis to be my own work, he steered me in the right the direction whenever he thought I needed it.

I would also like to thank co-supervisor Dr. Aysar Al-khalidi, IT Director of the University of Sulaimani UoS as the second reader of this thesis for his continuous support and encouragement, his valuable guidance. He definitely provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

I would also like to acknowledge Dr. Alaa Alhadithy, head of Database Technology department of the Technical College of Informatics TCI at Sulaimani Polytechnic University SPU, I am gratefully indebted to his very valuable comments on this thesis.

I must express my very profound gratitude to my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Finally, I am greatly honored to graduate from NEU.

Gulala Hama Amin

To my family…

ABSTRACT

Smartphone technology has made tremendous strides in the field of electronic devices. Since they have become an integral part of the individual's life, it has become an instantaneous necessity in human life; which shorten the distances, helped to access the information and deliver to the world in a few moments with a reasonable price.

Mobile phones have a specific memory size compared with other devices such as laptops or desktop computers. So, running multiple applications; at the same time; consume the memory and negatively affect the mobile performance. Therefore, this feature makes software developers take into account develop applications that consume a small amount of memory.

Nowadays, sharing service applications become more and more popular on different mobile device platform, because it helps to decrease the memory usage. The main idea of this thesis work is sharing services by designing and implementing a client server application that can run on the same mobile device. The application has been developed for Android mobile operating system or any Android tablet using inter-process communication (IPC) and android interface definition language (AIDL).

Keywords: AIDL; Android client server application; IPC; sharing services; smartphone application

ÖZET

Akıllı telefon teknolojisi, elektronik cihazlarda muazzam bir sıçrama yapmıştır. Kişisel hayatımızın vazgeçilmez bir parçaları olmalarının yanı sıra, mesafeleri kısaltan, bilgiye ulaşmamıza yardımcı olan ve uygun fiyatlarla anlarımızı tüm dünyaya yaymamızı sağlayan bir gereklilik haline gelmişlerdir.

Hareketli telefonlar, dizüstü ve masaüstü bilgisayarlara oranla daha belirli hafıza boyutlarına sahiptir. O yüzden, ayni anda çoklu uygulamaları çalıştırmak hafızayı tükettiği gibi performansı da düşürmektedir. Bu yüzden, yazılım geliştiricileri düşük miktarda hafıza tüketen uygulamalar geliştirmeye çalışmaktadır.

Günümüzde, farklı cihazlardaki platformlar üzerinde servis uygulamalarını paylaşmak daha da fazla popüler hale gelmiştir çünkü bu, hafıza kullanımını azaltmaya yardımcı olmaktadır. Bu tezin başlıca amacı, servislerin paylaşımını, aynı hareketli cihazda çalıştırılabilecek bir istemci sunucu uygulaması dizaynı ve uygulaması geliştirmektir. Uygulama, Android işletim sistemi ve süreçler-arası iletişim (IPC) veya Android arayüz tanım dili (AIDL) kullanan herhangi bir Android tablette kullanılmak üzere geliştirilmiştir.

Anahtar Kelimeler: AIDL; Androind istemci sunucu uygulaması; IPC; Servis paylaşımı; Akıllı telefon uygulamaları

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

GPS	Global Positioning System
OS	Operating Systems
AIDL	Android Interface Definition Language
IPC	Inter Process Communication
I/O	Input/Output
DVM	Dalvik Virtual Machine
UI	User Interface
API	Application Programming Interface
XML	Extensible Markup Language
RPC	Remote Procedure Calls
SDK	Software Development Kit
IDL	Interface Definition Language
URI	Uniform Resource Identifier
RSA	Rivest Shamir Adleman
IDE	Integrated Development Environment
ADT	Android Development Tools
JDK	Java Development Kit
JDK	Java Development Kit
RMI	Remote Method Invocation

CHAPTER 1

INTRODUCTION

In the last decade, wireless communication manufacturers grown in a very quick way; this made wireless communication one of the fastest rising technology sections in the world. Nowadays, using smartphones are increasing very swiftly, the reason behind that is the importance of their functionality in everyday life, which represents the new boundary to access the internet and the World Wide Web. Also, people have become more and more needy on the information accessible on the internet, and they more want to get access to the internet services, not only from their home and office computers, but also from their mobile devices (Isakow and Shi, 2008).

Moreover, in the last twenty years, mobile content; multimedia applications seen and used in mobile phones such as different electronic games, city guides, video downloaders, images editors, location navigators, and applications that have multitask characteristics; has become increasingly important worldwide. Smartphone users became able to exchange messages, pictures, reserve medical checkup appointments, exchange vouchers, get directions of roads, and check for cheap flight offers and surfing the net. On the other hand, mobile device market offers several competing mobile hardware and software such as Windows mobile from Microsoft, iPhone OS from Apple, and Android from Google. (Schreiber, 2011).

1.1 Android Services

Services could be defined as components which they are run in the background without a straight contact with the user. Since services have no any interaction with the user this leads to be some kind of restrictive to the activities' life cycle of the application. Generally, a service is useful for processes that are on the running situation and repetitive stretched type, for instance, downloading multiple resources from internet, checking the incoming messages in the email inbox, processing data.

Service run and bound with an advanced precedence that are invisible or inactive activities and thus it is simply terminated by the Android operating system. It is worth mentioning

that services can also be formed to be enabled restarted if they ended by the Android operating system whenever appropriate system resources are available or accessible again. It is potential to assign services the same priority as foreground activities. In this case it is essential to have a visible notification active for the associated service. It is often used for services which play videos or music (Vogel, 2014).

1.2 Sharing Services

A Service is an application element which achieves long-run processes that are invisible in the background, so it is not necessary to have a user interface. During the processing time, another application program can start a service, stays running in the background even if the user shifts or moved to another application service. Also, a component can bind and reserve a service to interact and communicate with it using inter process communication IPC mechanism. For instance, a service can handle different network transactions, listen to music or songs, perform file transfer, or interact with a content provider, all from the background (Android Developers, 2017).

1.3 AIDL Concept

Android Interface Definition Language or AIDL for short, handles the interface necessities between a client application and a service application so both can communicate at the same level through inter process communication or IPC. All the objects in the process breaking down into primitive data that Android can recognize. This is a very important required part, simply because a process of a specific application cannot access the memory of the other applications. It is important to mention that AIDL supports only the following data types:

String data types, charSequence, Lists, Maps, and all native Java data types like integers, long integers, characters and Boolean which is used in logical propositions (Guru, 2017).

1.4 Aims of this Thesis

The aim of this thesis is to design and implement server and client objects to interact with remote services using AIDL.

This thesis studies the Android system's capabilities in developing applications that communicate with each other and provide services to other applications. A simple

application has been developed using the client and server architecture. The Client component implements user interface of the application and Server component implements Services and processes the client's request. The Inter Process Communication (IPC) is used for the client and server communication.

This application uses Android Interface Definition Language (AIDL) to implement and use remote methods. This work also addresses the concept of Binder and Intent in general, which are different levels of abstraction of IPC mechanisms.

1.5 The Importance of this Thesis

This thesis concerned with developing android mobile application software using android eclipse and java programming language.

The proposed system in this thesis is to design and implement a client server application that enables the client to send a request to invoke a specific process which is exist on the server. The proposed system is implemented through using IPC and AIDL to invoke remote methods. Through this procedure the system will need less memory space to accomplish process, since the server will be operated (run in background) whenever received a request from the client.

1.6 Thesis Structure

This Thesis consists of the following chapters:

Chapter one: Introduction

This chapter covers the importance of smartphones in every individual's life and sharing service between applications in these smartphones, also brief concepts of Service, AIDL and aim of the thesis and the importance of the thesis.

Chapter Two: The Android Platform

This chapter is intended to explain in general term what is Android platform and focus on the Inter Process Communication (IPC) and the Android Interface Definition Language (AIDL). Also, this chapter includes the environments, programs, and tools that used to design and implement the proposed system.

Chapter Three: Design and Implementation

This chapter includes the main design and implementation of the proposed system. Also, it includes the main point for implementing the system.

Chapter Four: Conclusion and Future Work

This chapter will present the powerful points surrounding the Android and sharing the Android applications using AIDL. Moreover, it includes the main ideas behind the proposed system are concluded through this thesis.

CHAPTER 2

THE ANDROID PLATFORM

Android is one of the most popular mobile operating system in the technology arena. Developed by Android, Inc. The Google firm decided to buy Android, by 2015 the decision came into force, and it took over the development staff as well. This giant company decided to change the policy of its' new company by making it to be open source and permitted codes for free. So, under the open source Apache License, an extremely amount of Android code was released. Making Android open source enable users whom desire to use Android freely download the complete Android source code for use. Furthermore, hardware companies can enhance new feature extensions to Android and they can modify the operating system to produce unique output from other products. This has reflected its impact on vendors, making them more receptive to this new model. This is exclusively true for firms under the effect of iPhone's operating system, such as Sony Ericsson and Motorola, whom own their developed mobile operating systems for many years. Among of these companies there are some of them which had to challenge to find different methods to renovate their products, when the iPhone was launched. In addition, most of those firms think Android is a solution, and using android operating system makes them to continue in designing their own hardware and software, which powers it. One of the main feature of using the Android operating system is that, this system suggests an integrated method to application development. It's a great achievement for developers to be able to run their applications on different devices types. One of most significant portions of the success are applications, in the arena of smartphone industries and new technology trends. Thus, the smartphone industry firms realized that Android operating system is the best optimism to challenge the onslaught of the iPhone, the reason is that a large base of applications commands by iPhone (Lee, 2011).

2.1 Android Versions

Like any other successful companies, Android has many releases and numbers of updates. The Table 2.1 shows many versions of Android, their released date and name of code (Lee, 2011).

Table 2.1: Android versions and codenames (Lee, 2011)

Code Name	Version Number	Initial Release Date
Not available	1.0	23th of September 2008
Not available	1.1	9th of February 2009
Cupcake	1.5	27th of April 2009
Donut	1.6	15th of September 2009
Éclair	2.0 – 2.1	26th of October 2009
Froyo	2.2 – 2.2.3	20th of May 2010
Ginerbread	2.3 – 2.3.7	6th of December 2010
Honeycomb	3.0 – 3.2.6	22th of February 2011
Ice Cream Sandwich	4.0 – 4.0.4	18th of October 2011
Jelly Bean	4.1 – 4.3.1	9th of July 2012
Kitkat	4.4 – 4.4.4 – 4.4W – 4.4W.2	31th of October 2013
Lollipop	5.0 – 5.1.1	12th of November 2014
Marshmallow	6.0 – 6.0.1	5th of October 2015

2.2 Android Architecture

The Android operating system workings in different Layers, in order to understand these layers as shown in Figure 2.1, these layers are illustrated which make up the Android operating system (Lee, 2011).

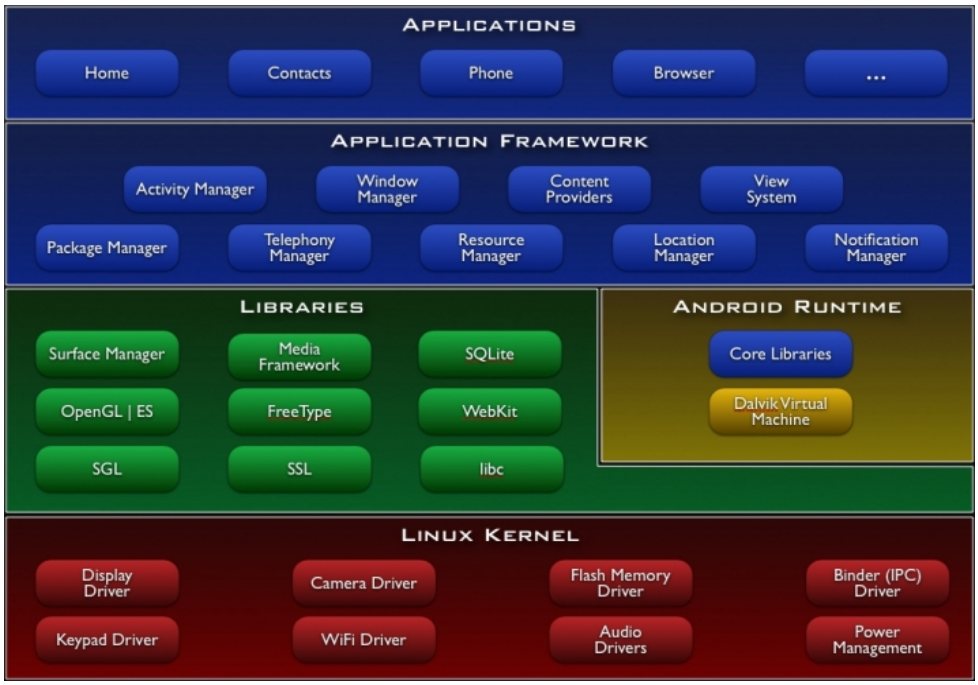

Figure 2.1: Android architecture (Lee, 2011)

From Figure 2.1, it is clear that the Android stack is divided to four major layers; Applications, Application framework, Libraries and Linux Kernel, from up to down.

2.2.1 Linux kernel layer

The Android operating system is based on this kernel. The Linux layer makes the interface between the software layer and the hardware layers. At this layer, the management of memory, the management of user process, the management of network commutation and driver management, security management, are handled.

2.2.2 Android runtime and core library layer

The job of this layer is providing a Dalvik Virtual Machine (DVM) and the core set of libraries, which is important to run the JAVA applications. The Dalvik VM is responsible

to execute the .dex (Dalvik executable) files. It is optimized to run the small footprint of applications at quick and fast speed.

Each of application runs its own process on Linux kernel, and it runs a different instance of Dalvik VM.

The Android operating system root base contains a set of library packages of C++ and C programming languages the Android component uses them. These packages can be accessible by developers using standard application frameworks. In the below Table 2.2, description of each library can be seen.

Table 2.2: Android runtime and core library layer (Android Developers, 2017)

Library	Description
C System Library (libc)	Improved library for embedded devices
Media libraries	image files, MPEG4, JPG, and PNG, video and Audio files
Surface Manager	arrange displaying subsystem
LibWebCore	Web browser engine to manage the web views
SGL	2D graphics engine
3D Libraries	OpenGL ES 1.0 3D libraries for high quality 3D raster graphics
Free Type	Bitmap and Vector Font Rendering
SQLLite	Lightweight relational database with SQL access

2.2.3 Application framework layer

This Layer provides a set of frameworks to and core functions to create and manage the user interface, run background jobs, set notifications and alarms. The components re-usability provides a flexible application development with framework layer. The application architecture allows the application to publish its features to other applications.

The android operating system provides the following set of system services:-

- Set of rich and extensible views used to build the user interfaces for applications.
- Content Providers, which uses to share and access the data between applications.
- Resource Manager, which uses to manage the resources.
- Notification Manager, that uses to display the alerts and notifications on status bar.
- Activity Manager, which uses to manage the lifecycle of application and state management.

The top layer is the application layer where lot of applications is bundled with platform and developers can build their own applications (Lee, 2011).

2.3 Android Application Components

This is the vital part of an Android application. The file with .xml extension, called manifest file of the application, defines the component of the application and the way they communicate.

These components can be used within an Android applications. As shown in the below Table 2.3 and Figure 2.2 (Sygida et al., 2016):

Figure 2.2: Android application component (sygida et al., 2016)

Table 2.2: Android application components (Android Developers, 2017)

Component	Description
Activity	Command the user interface and manage the user communication to the device screen.
Services	Manage processing in background related to an application.
Broadcast Receivers	Manage interaction between applications and Android OS.
Content Providers	Database management topics.

2.3.1 Activities

Activities are among the fundamental parts of applications on the Android operating system. They serve like access point for a users' interaction with an applications, and they are also central how a user routes within applications or between applications.

Each Activity contains at least three of the following events:

- **onCreate():** during the creation of the activity is called.
- **onStart():** for activities that are in visible state to the user, this method is called.
- **onResume():** once the user starts communicating with activity this method is called.
- **onPause():** it starts in case of present activity paused by user and prior activity is resumed.
- **onStop():** this starts in case the activity is in hidden state to the user.
- **onDestroy():** it is called before the system terminate the activity (the system may terminate it to protect the memory, or terminated manually) .
- **onRestart():** for a stopped activity, this method used to restart it again.

In Figure 2.3, the diagram of android life cycle of an activity and different steps, from the beginning of the activity's running start to the end or destroy state (Griffiths & Griffiths, 2015).

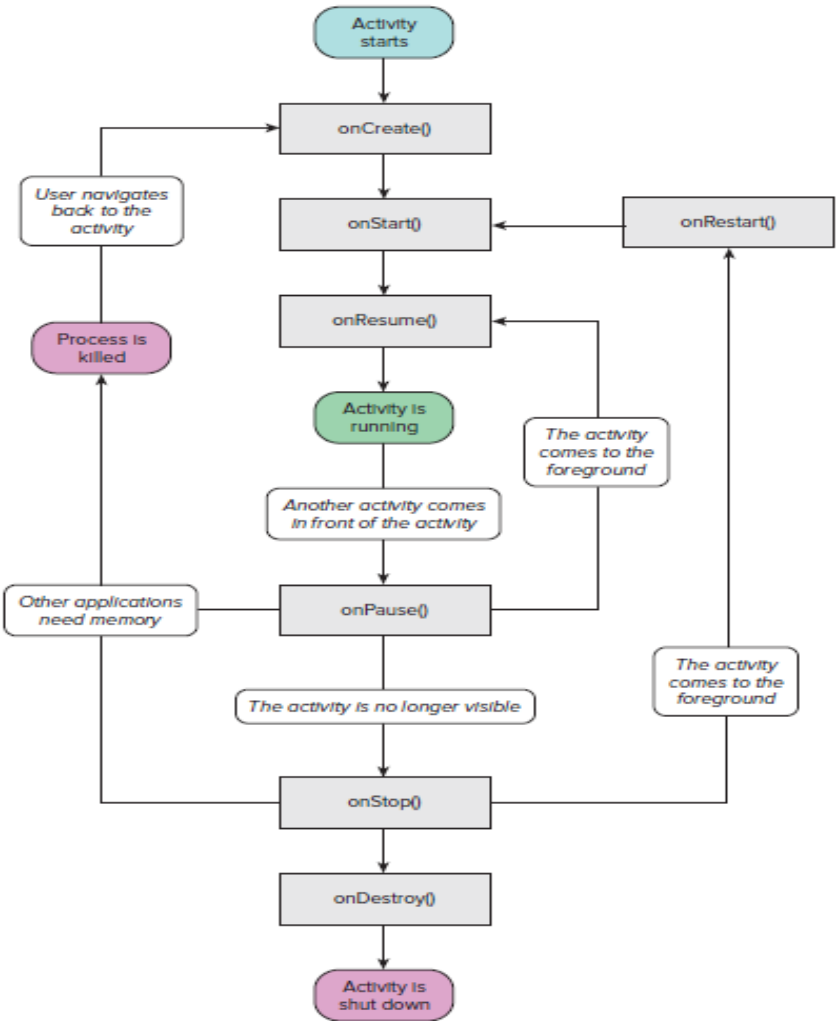

Figure2.3: Android application life cycle (Lee, 2011)

2.3.2 Android application services

Services are applications in Android that run in the background without any interaction with the users. For instance, when the user in the process of using a specific application, he/she might want to play songs in the background simultaneously. In this situation, the service doesn't necessary to communicate with the user. Also, services are perfect for states; in which there is no need to present a user interface to the user. For instance, an

application that repeatedly takes the location of the user. In this case the programmer can write a service to do that in the background (Lee, 2011).

A service is similar to Broadcast Receiver and Activity. It can be started independently of its Intent Filters by specifying a Component. Also, services can also be protected by adding a permission check to its service tag in the Android manifest. The Binder interfaces is able to check permissions on services caller, which permitting to implement several permissions at the same time or different permissions on several applications and different time. Consequently, a service offers many techniques to ensure that the caller is reliable, which is similar to Activities, BroadcastReceivers and Binder interfaces.

Android services contains of three different types (Darcey, 2012):

2.3.2.1 Scheduled

Services are scheduled when an API such as the JobScheduler, introduced in Android 5.0 (API level 21), launch the service. A user can use the JobScheduler by registering some jobs and specifying their requirements for network. Then, the system; gracefully; schedules the jobs for execution at the correct times. (Android Developers, 2017).

2.3.2.2 Started

A Service starts when one of the application component; for example activity; calls the method startService(). When it is beginning to start, this service indefinitely can run in the background, even when components which started it had been destroyed. Normally, startService() method accomplish single process and do not send back the result to the caller of the service. For instance, a service may upload/download some files from the net, and the service should stop itself, when the operation is complete (Android Developers, 2017).

2.3.2.3 Bound

By calling the bindService() method for a specific service, this service becomes bound when an application component binds to it. A bound offers the interface for the client side and the server side, which permits components to communicate with the service, send requests, and get results with inter process communication (IPC). Bound services run only as other applications component are bound to them. At once, multiple

components can bind to the service. On the other hand, the service is destroyed, when all of them unbind (Android Developers, 2017).

2.3.3 Broadcast receiver

It is another android application component; this component is replies to system varied broadcast announcements. There are many broadcasts patent from the system such as, an announcing broadcast when the screen has turned off, another announcing is when the battery is in low charge state. It is worth mentioning that applications also can start their own broadcasts such as, different applications will be notified that some data has been downloaded to the device and is now accessible for them to use. Usually broadcast receiver do not show the user interface, by creating a status bar notification it will alert the user whenever broadcast events occur or happen (Android Developers, 2017).

2.3.4 Content providers

This manages access to a central source of data. Content providers of an Android application and regularly provides its user interface to function with the data. Anyway, content providers are mostly expected that other applications use it which access the provider using a provider client object. Providers and provider clients offer a consistent, standard interface to data. Also, it manages inter process communication (IPC) and also manages the process of accessing data securely.

Normally, content providers in two situations will be used or users deals with them; if they want to implement code to access a present content provider from another application, or they might want a new content provider being created in their application to handle sharing of data and resources among applications. (Android Developers, 2017).

2.4 Android API

It is a number which identifies the framework Application Program Interface (API) revision offered by a version of the Android platform.

The Android platforms provide a framework API that applications can use to interact with the underlying Android operating system:

- Classes and core set of packages.

- To assert the manifest file of the application, which is a set of XML elements and features.

- To assert and access all the resources of the application, which is a set of XML elements and features.

- A number of Intents.

- A number permissions in which the application can request, also permission implementations involved with the system.

When a new API version will be introduced, it stays compatible with earlier versions of the API. Since updates to the framework of API are designed, the compatibility will be possible. So, most changes in the API are introducing new feature and functionality. The older replaced parts are disapproved but are not deleted, so that existing applications can still use them. Although such changes are only needed to ensure API robustness and application or system security, in some cases, parts of the API may be improved or deleted. The framework Application Program Interface which an Android platform delivers is specified using an integer identifier called API Level. Each Android platform version is supporting an API Level (Android Developers, 2017).

When application developers intended to design and develop an application program they must take into account two things for choosing an API level for the application:

a) Distribution Issue: This means that in case the application was for API level 12, it is not running on API level 11 and earlier, this results in reducing the number of devices can run this application. So the number of devices support the developed application is a very significant point.
b) Functionality Limitation: Choosing an API level of a lower type might support more devices but is likely to face a problem which is gaining less functionality for the developed applications. Application developers could work harder to reach features that they could have simply gained if they chose higher API level (Shah and Rahman, 2013).

2.5 Inter Process Communications

It is a set of techniques to change data among different threads in one or more than one process. Inter Process Communication method is divided to many types. Some methods used for passing messages and others for synchronization or shared memory.

The methods of IPC may depend on the latency and bandwidth of communication among the threads. Also, it may depend on the type of data communication. In addition, some of the main reasons to provide an environment which allows IPC are information sharing and security.

So, the most important feature of Android operating system is trying to eliminate the functions duplications in different multiple applications. Applications functionality must have few dependencies and must be able to other applications. The following Figure 2.4 shows the IPC mechanism in Android OS platform (Griffiths & Griffiths, 2015; Singapati, 2012).

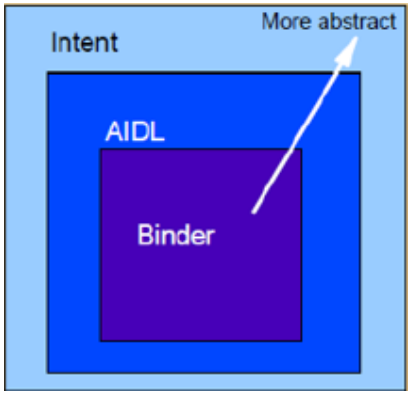

Figure 2.4: Abstraction of IPC in Android (Singapati, 2012)

2.5.1 Intent

The purpose of the intent is to provide a high-level system of IPC. Also, it is enabling applications to Service based on users' actions. In different words, it is not needed for hardcoded path to any application to use its functions and exchange data with them.

According to O'Reilly Media, Intent is responsible to provide the highest-level abstraction of IPC in Android platform. Furthermore, it delivers data among applications. Android application includes three components: activities, services, and broadcast receivers.

When the activity is activating, intent start to define the action to perform such as, opening contacts, or showing images. Activities can be starting to receive results or returns results in Intent such as when users select a phonebook contacts.

Intents have two forms. The first is the explicit intent addresses to a specific component. The second is an implicit intent to give the decision to the Android operating system. Moreover, if components are installed for one purpose, the android operating system will select the best component to run the intent (Singapati, 2012).

2.5.2 Binder

It is lowest level abstraction of Inter Process Communication in Android platform. Based on shard memory, it offers high performance.

Originally, it is developed by Be Inc. and later Palm. It is providing a richer high-level abstraction on the services of modern operating system.

In android system platform, a binder is a modified implementation of OpenBinder, which uses for everything, happens across processes in the core android platform (Jeong, 2015).

2.6 Remote Methods and AIDL

AIDL is responsible to resemble the remote procedure calls which are offering by another system. It is used to create rich application interfaces to support object based IPC among applications which are running using different processes. AIDL is making APIs remotely accessible. This thesis work is focused on using the AIDL, and it is explained in detail in the next following sections.

The AIDL is able to generate a Java class from the interface that uses for two purposes. First it is giving clients access the services by generating a proxy class. Second, it is generating a stub class which can be used by the service implementation.

The AIDL is generating the source code for clients and remote services in one file; which must be shared between remote service application and client application developers. On Android platform, one process normally cannot access and share the memory of other process. So, it needs to decompose the objects into primitives which will be understandable by the Android operating system. (Marko, 2017).

The AIDL interfaces are built by developers as same as Java RMI to automatically generate the invocation stubs (Sbîrlea, 2013).

When Android developers are building application which contains the .aidl file, an IBinder interface is generated by the Android SDK tools created on the .aidl file in the application project's generated folder in the file directory. When binding to a service occur by applications; then, call methods from the IBinder to perform Inter Process Communication (Android Developers, 2017).

Creating and using AIDL includes three steps, which are:

2.6.1 Creating AIDL interface definition

AIDL has an interface which is defined in an (.aidl). It must be saved in both the server and client application source codes. Each (.aidl) file defines only one interface. AIDL is supporting different types of data such as: the primitive data types in java programing; integer, Boolean, float, char etc. Also String and character values, map objects, List objects classes that implement Parcelable.

2.6.2 Implement the interface

Depending on the (.aidl) file, the SDK tools generate the required interfaces. Each interface has inner abstract class named (Stub); which responsible to extends Binder. A developer has to extend the Stub class and implement the required methods.

When a developer builds an application, a (.java) interface is created; which includes a subclass called (Stub).

2.6.3 Expose the interface to clients

When a developer is implementing the interface for any service, it needs to expose interface to clients which enables them to bind to it.

When clients connect to a service, the clients should be able to access to the class interface. So, if the clients and services are located in separated applications, the client's application must have a copy of the file with .aidl extension in the source folder in the directory file (Android Developers, 2017).

It is worth mentioning that, in addition to java programming, the aidl file can be constructed using C++ programming language, with some difference which is that, in case of using C++ programming language, C++ binder interfaces should produce the different level of similarity to the java codes equivalents written for the same application (Guanzhong, 2017).

2.7 Android Platform Security

Linux operating system is the kernel of the Android platform. Android platform uses with a wide range of electronic devices, such as cellphones and tablets. The Android operating system performance is depending on the processor capabilities. Since, security is an important issue of any hardware and software; it is a major part of Android devices. (Veracode, 2016).

2.7.1 Android data encryption

Using encryption algorithm on Android platform devices enables users to keep their files safe and secure even if their devices get stolen. In general, encryption is converting files to something different, and a decryption is converting them back to the original state.

So, using the encryption on android platform devices will help to encode all users' data. So, even if the mobile devices are stolen, the data will be unreadable (Android Developers, 2017).

2.7.2 RSA encryption\decryption in android

Rivest Shamir Adleman (RSA) is one of the best secure encryption algorithm which is currently use by many developers. This algorithm includes four steps; which are key generation, key distribution, encryption, and decryption. Also, it is asymmetric encryption by using public key and private key for encrypting and decryption confidential data. The public key is open for all people and it's mostly use to encrypting data.

The most complex part RSA is the public and private keys. Those keys use large prime numbers (p and q); which are generating using the Rabin Miller algorithm. A modulus n is calculated by multiplying p and q. This number is used by both the public and private keys. Moreover, it provides the link between them. Its length expressed in bits, and it is called the key length. The public key contains of the modulus n, which is calculated by

multiplying p and q; and a public exponent, e, which is normally set at 65537. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the toting of n (Techtarget, 2017).

2.8 Developing Android Application with Java

Java is one of the most popular language in programing for building and developing android applications. Since Java contains several of significant features as compared to other powerful languages, makes it to be a very good choice for application developers.

Java's main characteristic are:

- Very simple to understand.
- Has very robust and secure platform independent.
- Characterized as an object-oriented programming.

The java libraries, like: graphics, math, networking, and data structure libraries, are exist in Android SDK, which help developers in developing remarkable Android applications (Krishna, 2014).

2.8.1 Eclipse

Both eclipse and (ADT) Android Development Tools plugin, offers some important features for android application developers. Eclipse is an open source integrated development environment (IDE). It is; mostly; popular for Java development. Also the users or developers of different development platforms supported by the Android, like Windows operating system, Macintosh operating system, and Linux operating system, can download eclipse IDE from the eclipse foundation homepage (O'Reilly, 2004).

Eclipse has many available versions to download for free in their official website, and the one which chosen to be used in this work is Eclipse Mars 4.5.

2.8.2 Android SDK

SDK is an acronym for Android Software Development Kit. It is a set of tool kit which is used by application developers to develop applications for Android platform. The Android SDK contains these features:

- The libraries: It contain the libraries that are within the needs and requirements for developing an android application.
- The Debugger: In Eclipse there is debug perspective to control the debugging for the written java codes of an application.
- Emulator: In order to test an application program before installing it on an actual device, an emulator will be needed which a virtual mobile device is used for testing and running on computer.
- Android application program interfaces (APIs): The identification of API is very important for mobile application developers when starting to build any application project. This makes it easier to determine the range of devices that a specific application could be installed in, which is one of the important point in building mobile applications.

Whenever a new version of Android released by Google, an equivalent software development kit also will be released. This enables developers to write programing codes with the newest features. Hence, developers must install and download each version of SDK for the specific mobile phone.

It is worth mentioning that Integrated Development Environment (IDE) is the most common method for writing Android programs. The recommended IDE is Eclipse with the Android Development Tools (ADT) plug-in. However, other IDEs, such as NetBeans or IntelliJ, will also work (Alina, 2015).

2.8.4 Genemotion

Genymotion is a virtual machine, in another word an emulator which is very suitable for testing application, it can be used instead of the default Android emulator. Since Genemotion has many benefits because of the speed and achievement of this virtual

machine, it is a good choice for developers to test their application before installing it in a real device. (Genymotion, 2017).

CHAPTER 3

DESIGN AND IMPLEMENTATION

In this Chapter, the logical design and implementation of the proposed system will be described.

The first stage of this thesis is designing and implementing the server side application, which runs on an android mobile device. The purpose of this stage is to build a server which provides different and multiple services to the client side applications. The second stage of this work is to design and implement the clients' applications; which are access and use the server side services.

3.1 The Proposed System Architecture

The concepts of client and server are powerful functional abstractions. A server is simply a unit that provides a service, possibly to one or multiple clients simultaneously, and a client is a unit that consumes the service, as shown in the following Figure 3.1.

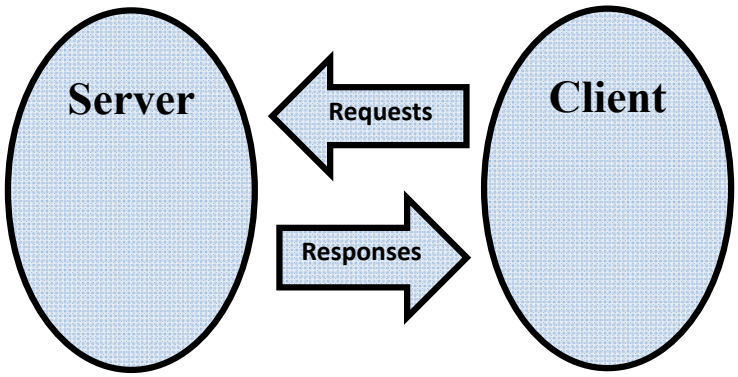

Figure 3.1: Client server diagram

Clients do not need to know the details of how the service is provided, or how the data they are receiving is stored or calculated, and the server does not need to know how the data is going to be used. The client-server architecture is a way to dispense a service from a central source. This proposed system includes a single server that provides services, and multiple clients that communicate with the server to consume its products. With the

proposed system, android users can use this application suit (client side and service side) after installing on their devices.

Users with this application send requests to the server service to receive the result back after execution specific functions on the server side. The following Figure 3.2 shows a very simple diagram illustrates the procedure of sending requests from a specific client to the service/server and getting back the result.

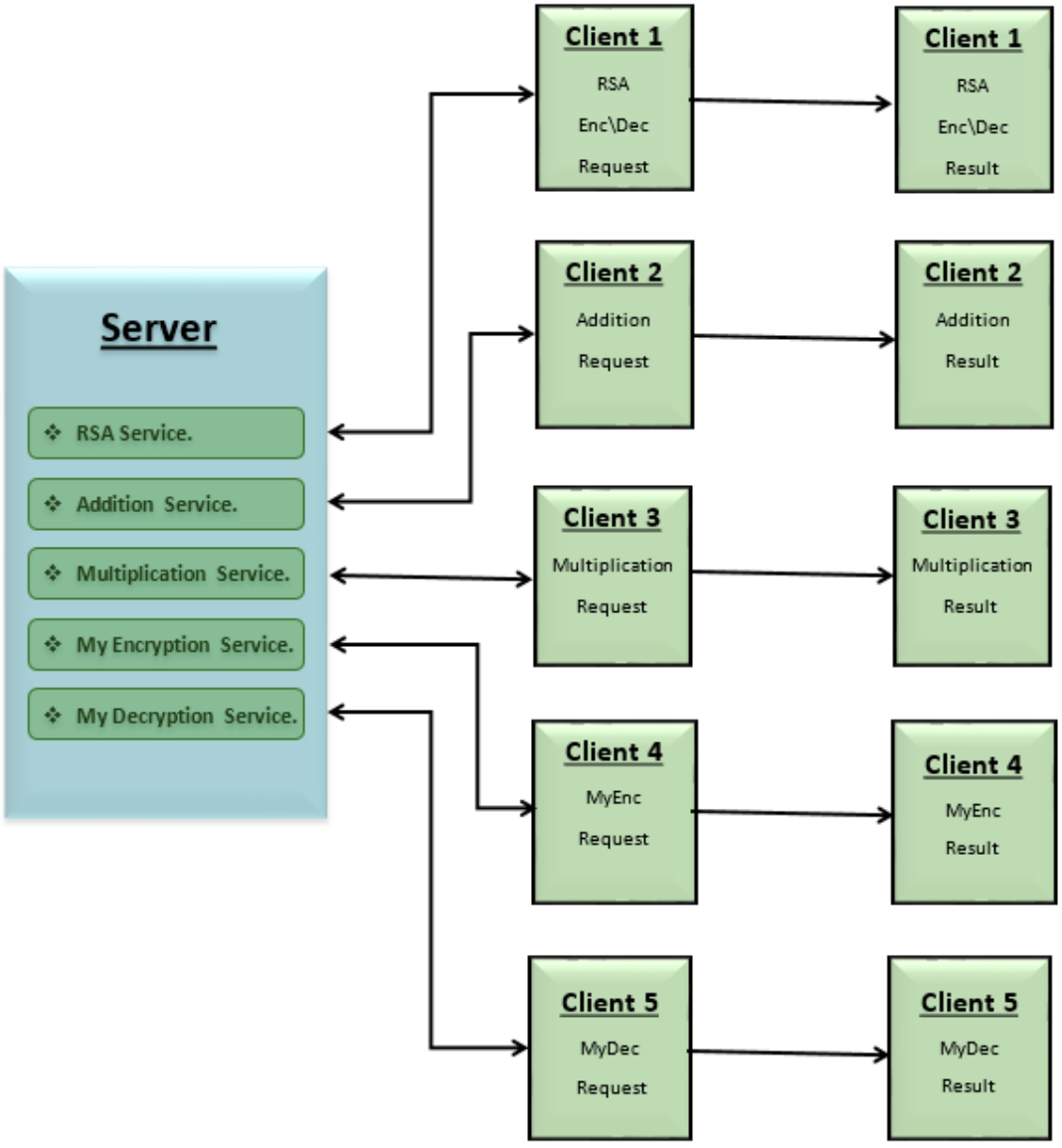

Figure 3.2: Proposed system main diagram

The above Figure 3.2 shows that the server and multiple clients have different jobs. The server's job is to respond to service requests from the clients, while a client's job is using the data provided in response to perform some task through AIDL.

Android interface definition language commonly known as (AIDL), with the inter-process communication (IPC), these android mechanisms enable activities to communicate in parallel with objects. In another word, the AIDL and binder provide powerful mechanisms for object-oriented inter-process communication that basically echo type method suggestions on java objects.

These mechanisms in the proposed system will be boosted in the term of a bound service application that uses the AIDL and binder. The main benefit of using them is to interact with a pair of bounds services for sending and receiving requests from a remote service by the client; as shown in the following Figure 3.3.

The Figure 3.3 below shows the main flowchart of this work. The start is the binging of the flowchart, and the number 1, 2, 3, 4, and 5 are the five applications; which were designed and implemented in the previous chapters.

Number 1 is the RSA application, number 2 is the Addition application, number 3 is the Multiplication application, number 4 is the MyEncryption application, and number 5 is the MyDecryption application.

Each application has to connect to the server to start its functions. Each one of those applications will be explain in detail in the next sections of this chapter.

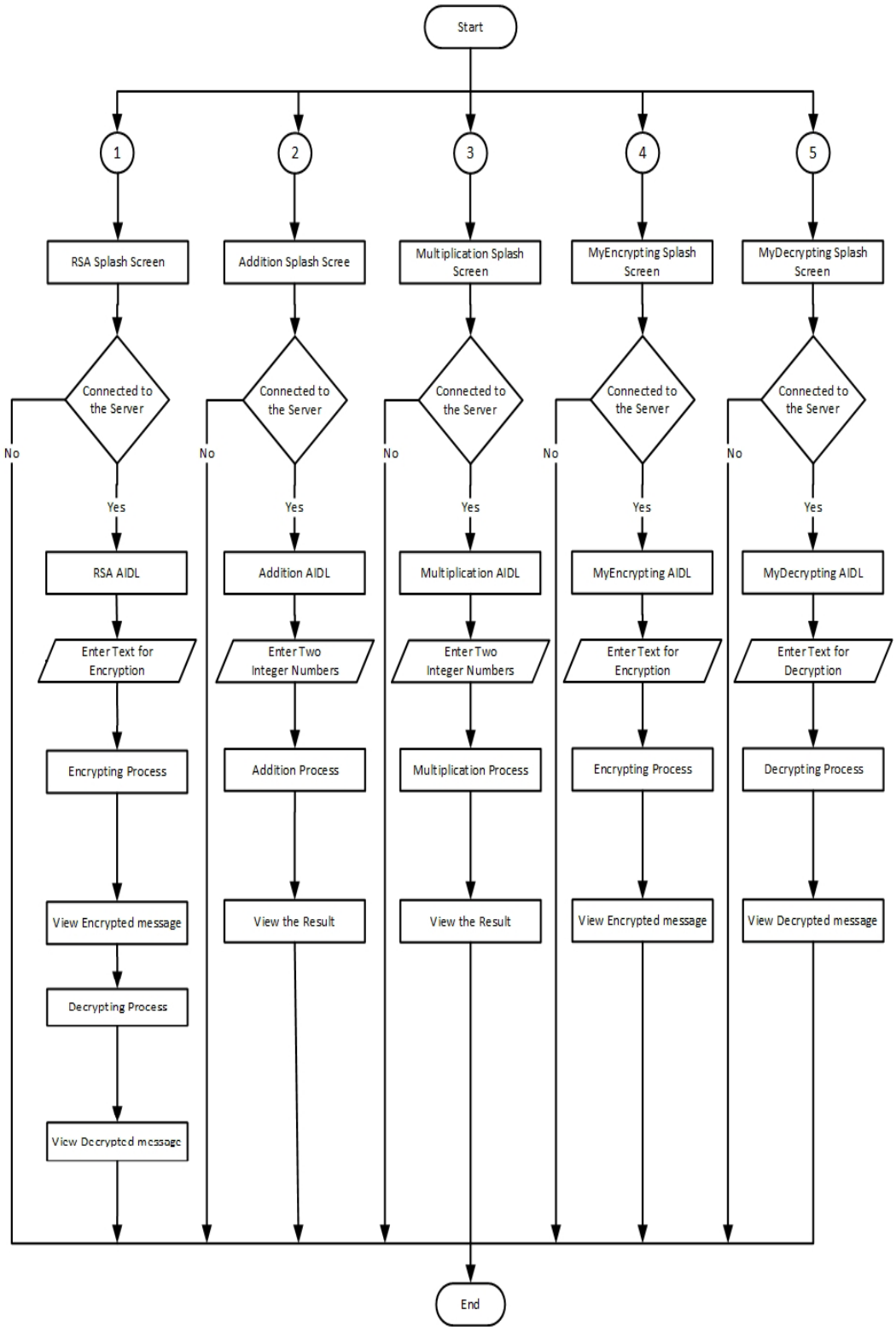

Figure 3.3: Proposed system flowchart

3.2 The Proposed System Design

In this part, the logical design of this work will be presented, which includes two parts:

- A server side application: it includes five functions and provides different services using AIDL, which are: RSA Service, Addition Service, Multiplication Service, MyEncryption Service, myDecryption Service.
- A client side application: it includes five separated applications, which are able to connect with the server and use the service; which are already build on the server side.

Both, the server and client connect and work on the same android device using AIDL. The reason behind using AIDL is that each android application (for security reasons) runs in its own process and cannot normally access the data of another application running in a different process; And to allow one application to communicate with another application running in a different process, the AIDL is used in this research work.

3.2.1 The server system design

The server side incudes five functions and services; which are: (RSA Service, Additions Service, Multiplication Service, myEncryption Service, myDecryption Service). At the beginning, the systems was designed and implemented just with a one server side service, which was RSA service for encryption and decryption messages. Then, to make this work clearer and show the advantages of using the AIDL, it is decided to build and add more services and function to the server side, which are: Addition service, Multiplication service, myEncryption service, and myDecryption service. These services are well designed and implemented to enable clients access for using them; and each part of these services will be described in detail in the next sections of this chapter, as shown in the following Figure 3.4:

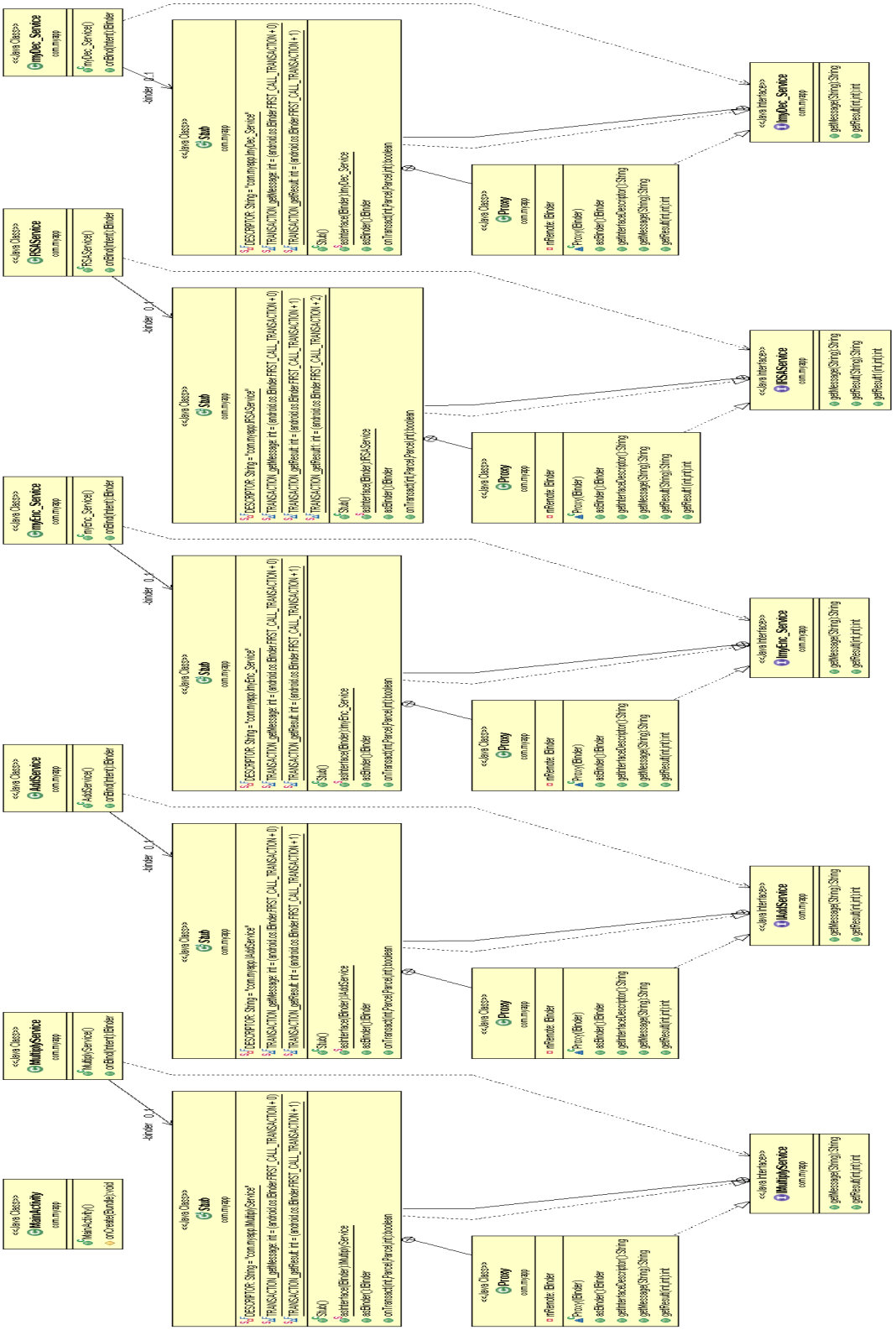

Figure 3.4: Java class diagram of the server side

Figure 3.4 shows the server side java class diagram. It shows the five different service which are developed in the same package with different AIDL. Clients can connect with each one of these services to run separately.

3.2.2 The client system design

The client system includes five different applications which are:

- RSA Service.
- Addition Service.
- Multiplication Service.
- MyEncryption Service.
- MyDecryption Service.

These applications are designed and implemented separately. Each client application has access to use the server side services and functions through the AIDL. It is important to assure that in this work another clients like subtraction service, division service, or any other necessary services can be added to bind to the service and request for the service, undoubtedly after writing the necessary codes in the both server side and the client side.

3.2.2.1 The RSA service design

As it was mentioned before, RSA is one of the most popular and widely algorithm which uses for securing data transmission. The most important characteristic of this algorithm is that the encryption key is public and differs from the decryption key; which is kept secret. In this part of the work, the RSA service on the client side is designed and implemented, which enables clients' applications to access the RSA service on the server side using AIDL.

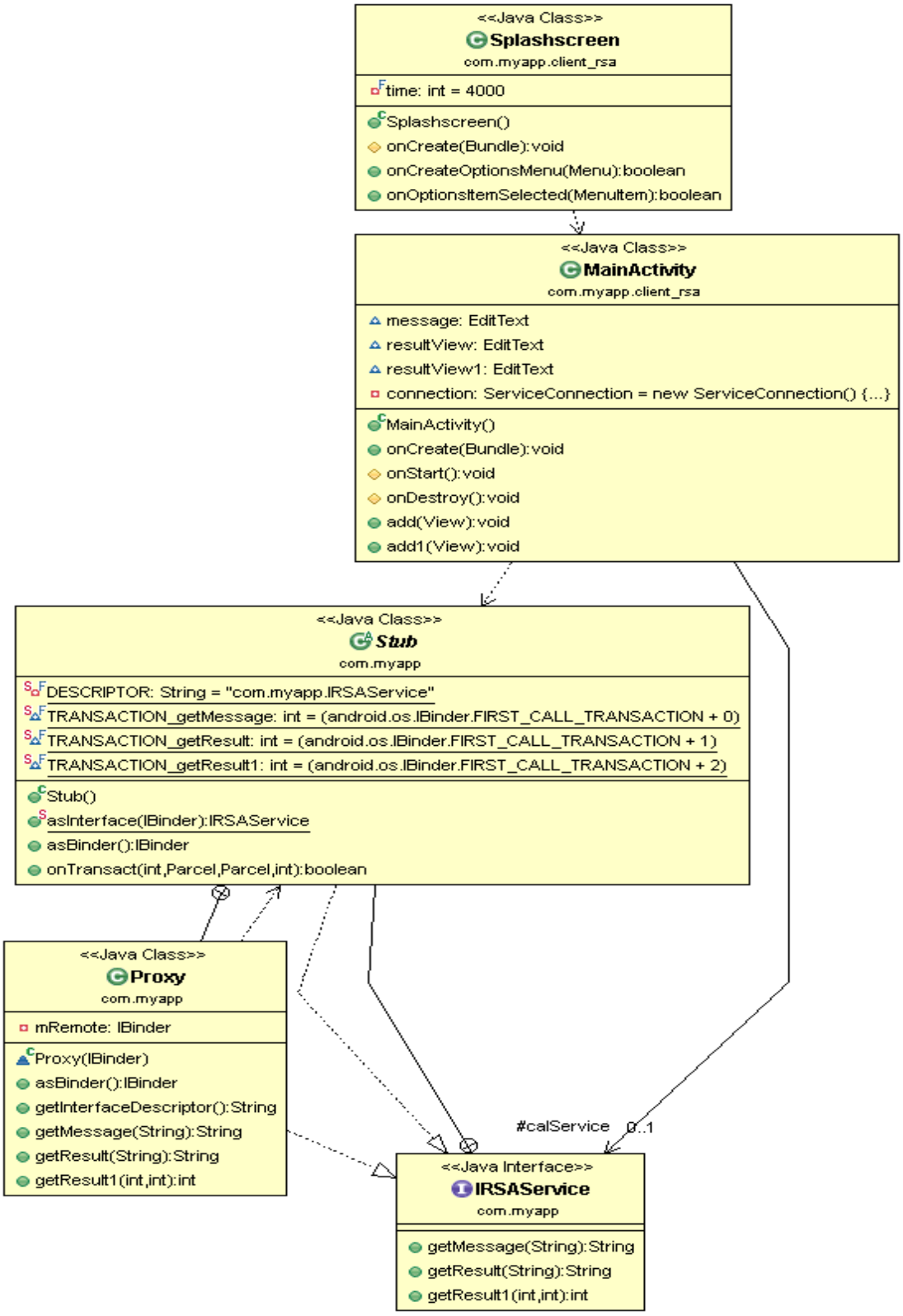

Figure 3.5: Java class diagram of the client side - RSA service

As shown in the above Figure 3.5, the service is designed using different classes and methods. The main activity of this application includes two EditText box. The first one is to enter the original message for encryption process, and the second box to get the encrypted message for decryption process.

3.2.2.2 The addition service design

The server side offers the addition function service for the clients. By this service, clients; which are needed to call and use this service; connect to server side. Designing this service needs to create multiple and different Java classes and subclasses; as shown in the below Figure 3.6.

3.2.2.3 The multiplication service design

The server side offers the multiplication function service for unlimited clients, as a calculator of two numbers. Every client; which needs to use this service; will make a connection using the AIDL. Designing this service includes many different Java classes and subclasses; as shown in the below Figure (3.7).

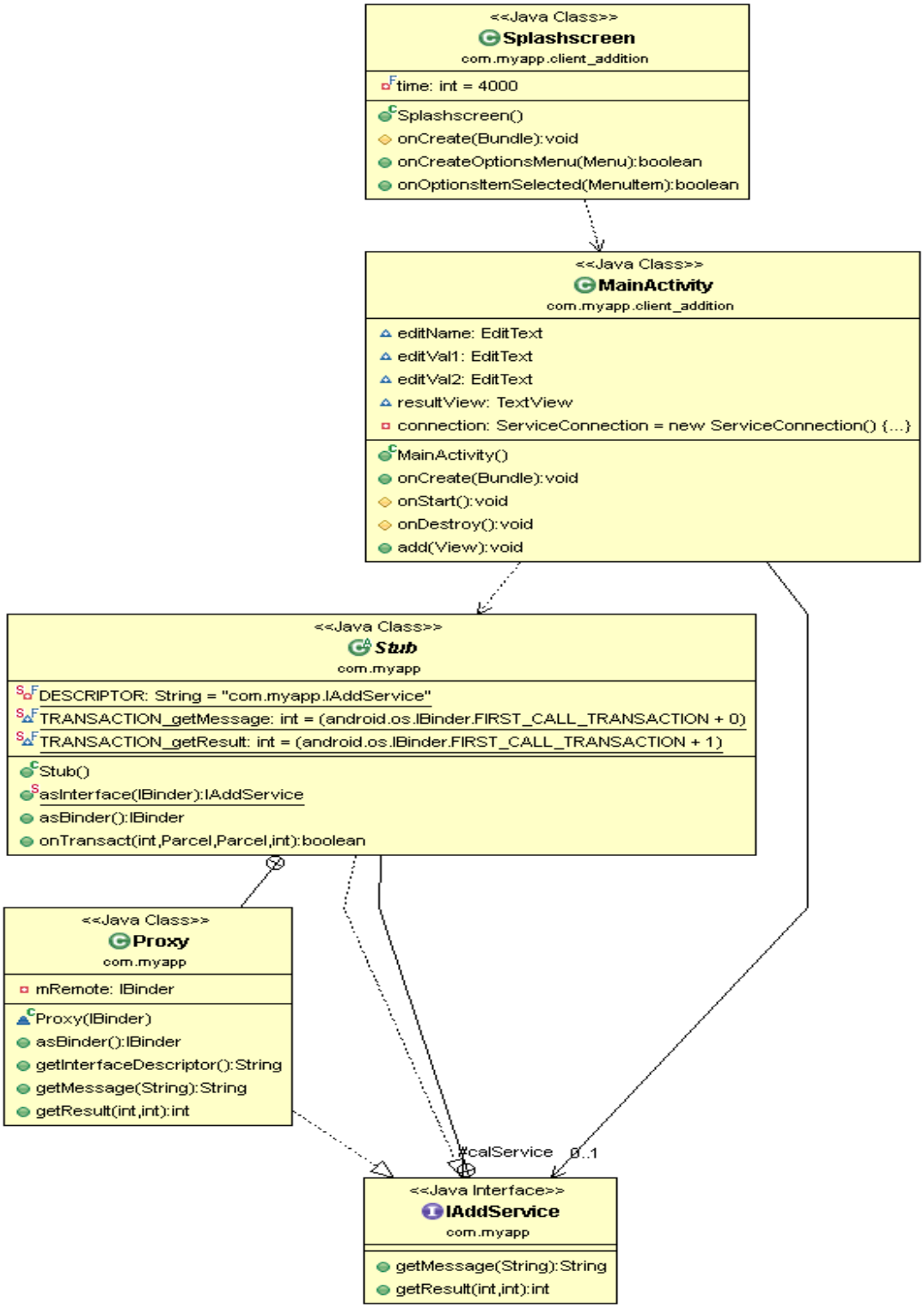

Figure 3.6: The client side - Addition Service

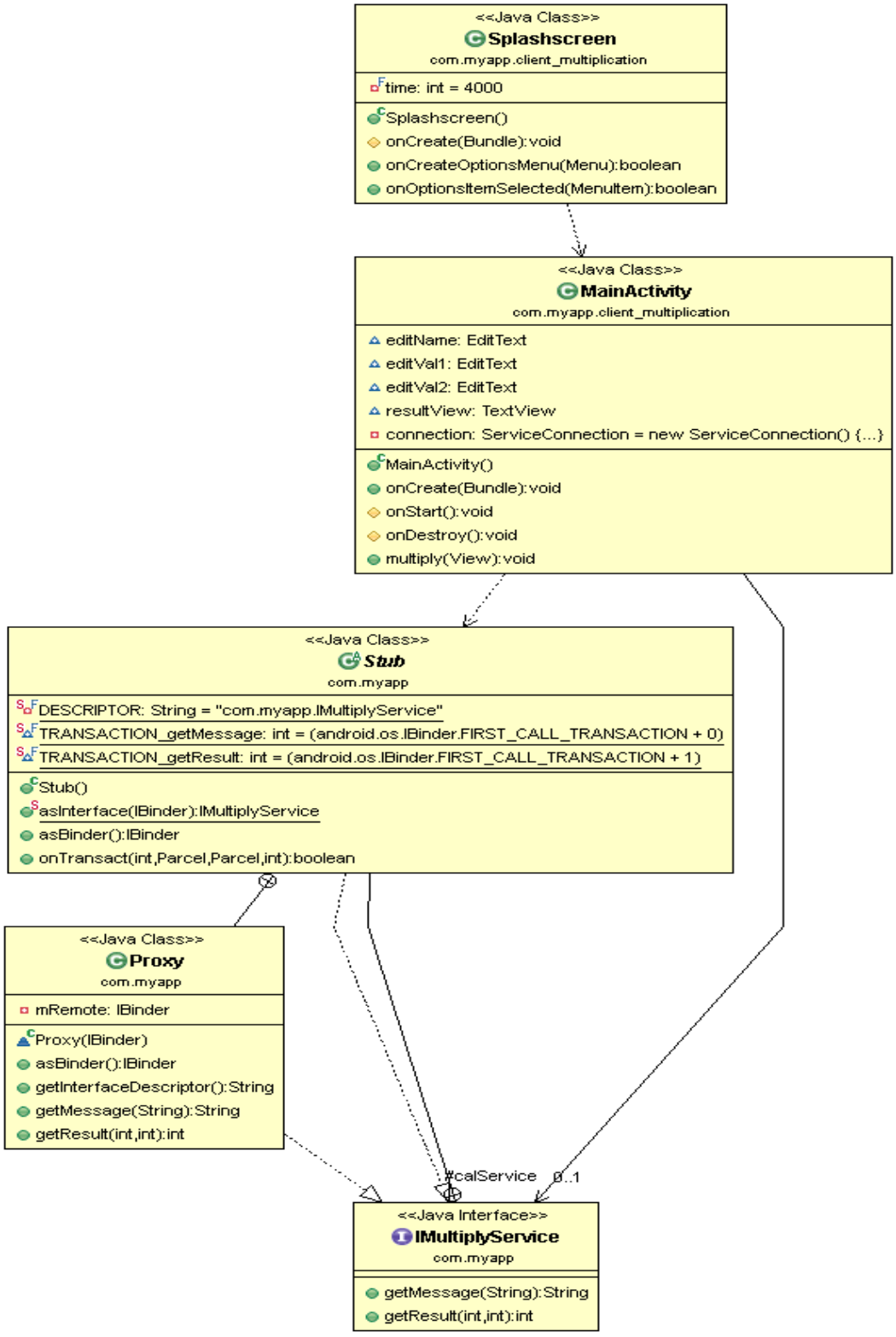

Figure 3.7: Java class diagram of the client side - Multiplication Service

3.2.2.4 The myEncryption service design

This part of the work is designed and implemented to encrypt messages of users. The reason behind naming (myEncryption) is using a personal simple encryption algorithm do develop this part of the work like the RSA; as shown in the following Figure 3.8:

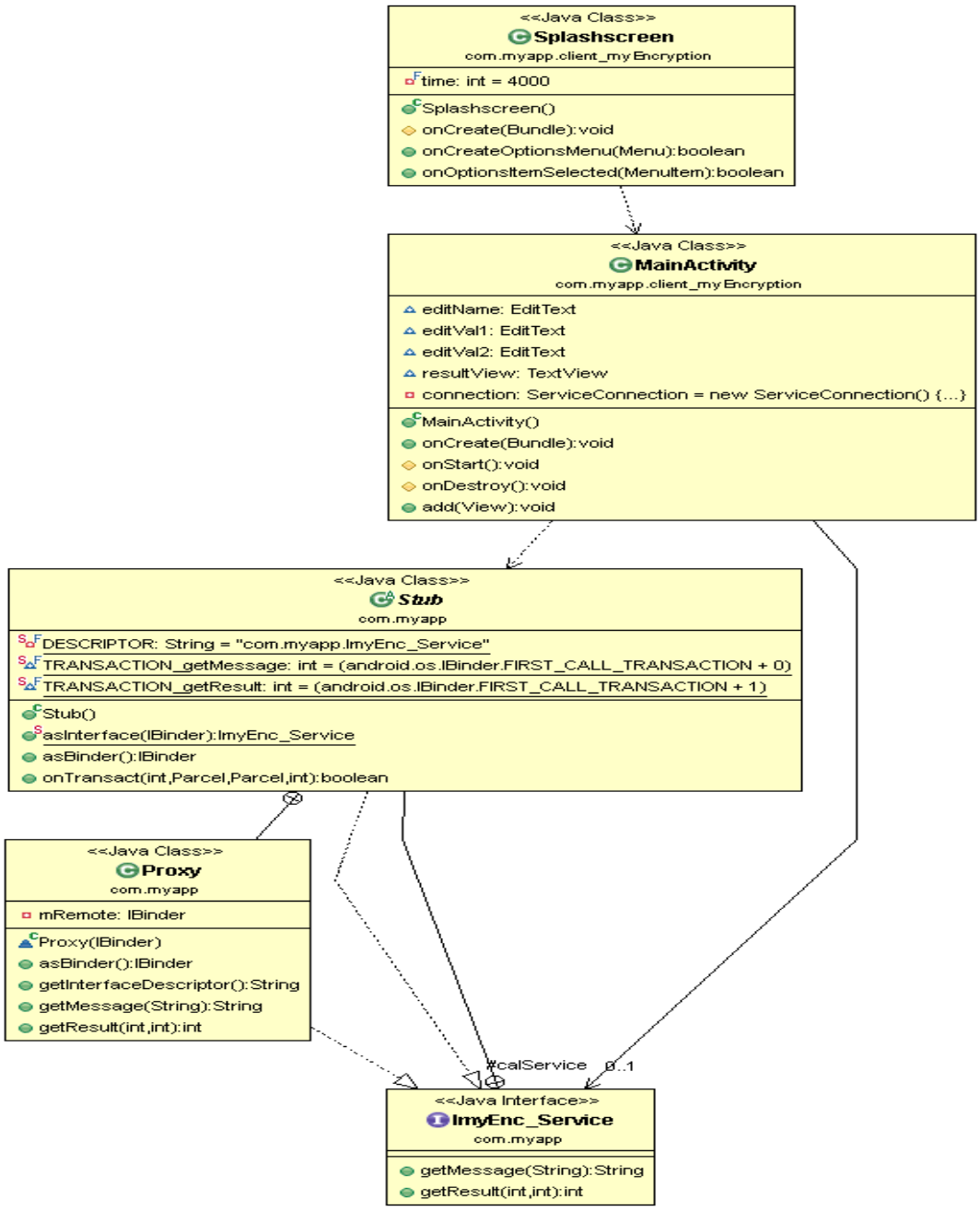

Figure 3.8: Java class diagram of the client side - MyEncryption Service

3.2.2.5 The myDecryption service design

This client application designed and implemented for decrypting the messages which are encrypted by the previous application (myEncryption application). In other words, client users can run the (myEncryption) application for encrypting messages. Then, use the (myDecryption) application for decrypting the messages. The class diagram for this application is shown in the Figure 3.9:

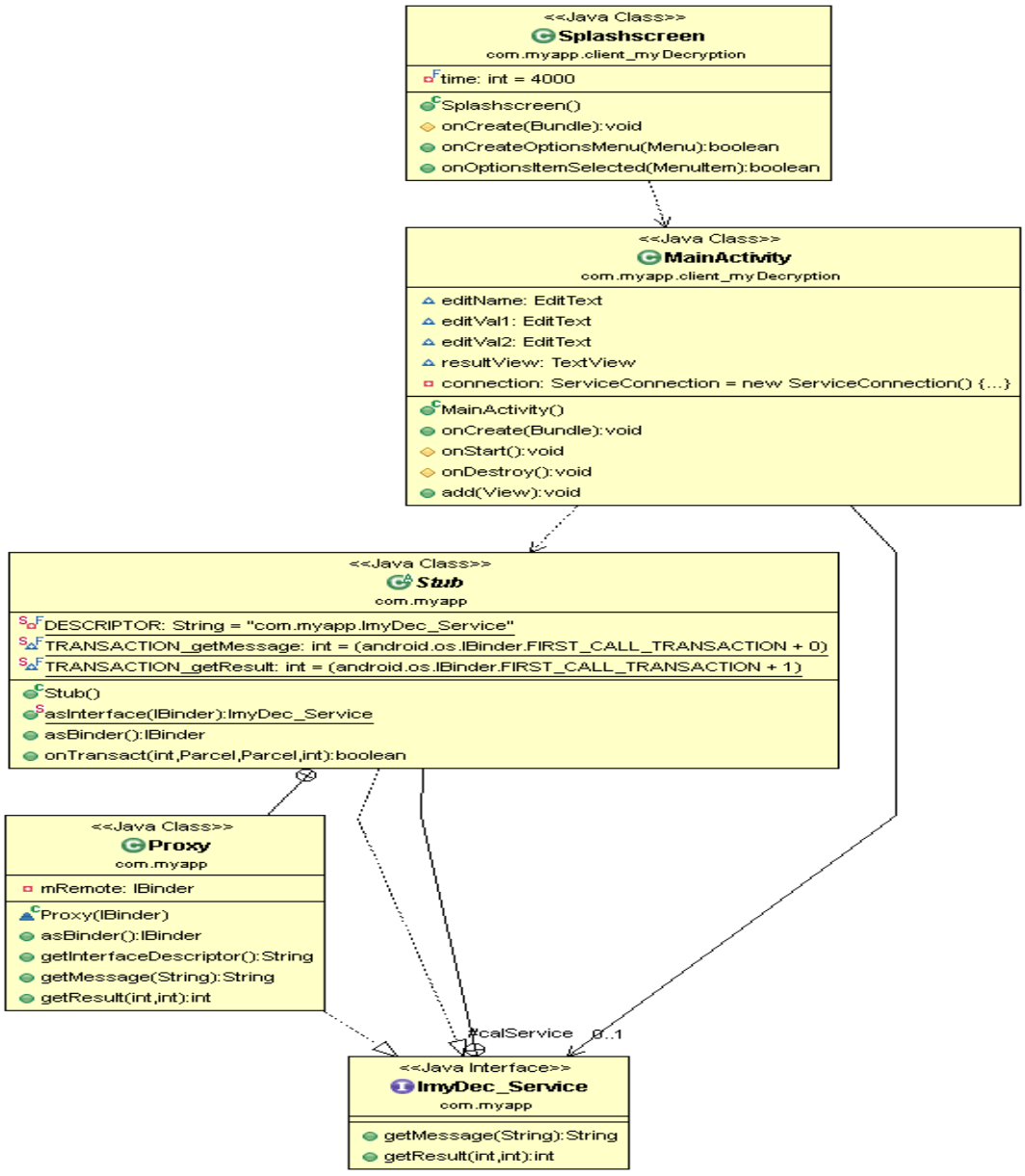

Figure 3.9: Java class diagram of the client side - myDecryption Service

3.2 The System Implementation

As mentioned in the previous chapter, java program language with some other tools is used to implement the practical part of this work. The server side services five different client applications; which are separately implemented and run on the same client. The implementation of each part will be explained in detail in the next sections of this chapter.
The Android Emulator Samsung Galaxy S4; which support API 17; had been used in the system implementation, as shown in the Figure 3.10 below:

Figure 3.10: Android emulator samsung galaxy S4

3.2.1 The server side implementation

After installing the server application on the android device, the application icon will be added on the home screen of the android device; as shown in the following Figure 3.11.

Figure 3.11: The client home screen

After running the (AIDL Server) application on the client machine, the server application will start and ready to serve the other client applications. When the server is successfully started with no errors, a short message which is “Server is ready” will appear on the main screen, as shown in the Figure 3.12 below:

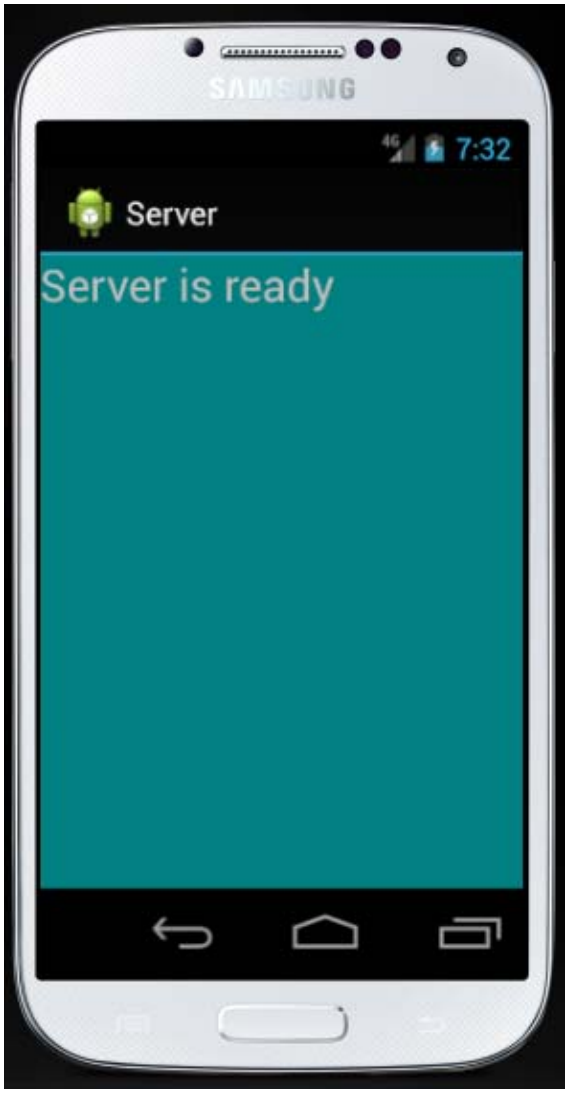

Figure 3.12: Running the server application

3.2.2 The client side implementation

As mentioned in the previous chapter, the client side includes five different applications which are:

- RSA Service.
- Addition Service.
- Multiplication Service.
- MyEncryption Service.
- MyDecryption Service.

These applications are separately implemented. Each client application has access to the server application to use the server side services and functions through the AIDL.

Figure 3.13: Clients icons in the home screen

The Figure 3.13 shows the five clients application after installation on the android client machine.

3.2.2.1 The RSA Service Implementation

Part one: After successfully implementing and running the RSA application on the android client machine, the Client will be ready to request the RSA service of the server application; which is run on the same Android device. Once the application will be run, a welcome splash screen will be displayed for 4 seconds, which is the first activity of the application, as shown in Figure 3.14:

Figure 3.14: Welcome screen of the RSA Application

Part two: after the first activity which was the splash screen, the second activity will be started, as shown in the following Figure 3.15

The second activity of the RSA application is connecting to the server side. To make sure that the application is successfully run and connected to the server, a short message

“Service Connected” will be showed on the screen for 2 seconds, as we can see in the following Figure 3.15.

Figure 3.15: Connection of the RSA with the server application

Part three: In the EditText, and as shown in the following Figure 3.15, the android user will be able to enter a text message for encryption using the RSA algorithm.

Figure 3.16: Entering a message for encryption

When the (Encrypting) button is clicked, the message; which already entered will be encrypted, as shown in Figure 3.16:

Figure 3.17: Encryption process result

In this part, the client has sent the message to server application. The server has done all the encryption process and encrypted the message. Then the server sent the message back to client to show as a result.

Part four: When the (Decrypting) button will be clicked the message that had been encrypted will be returned to the original message, as shown in the following Figure 3.17:

Figure3.18: Decryption process result

In this part and the same way, the decrypted message has sent to the server application. The server application has done all the decrypted process and decrypted the message. Then, send it back to the RSA client.

3.2.2.2 The addition service implementation

Part one: after installing and running the addition service application on the client machine, the client will start to connect and request the Addition service that is run the server side application.

Once the application is running, the welcome splash screen will be displayed for 4 seconds, which is the first activity of the application, as shown in Figure 3.18:

Figure 3.19: Welcome screen of the addition

Part two: The second activity of the Addition application is connecting to the server side. To make sure that the application is successfully run and connected to the server, a short message "Service Connected" will be showed on the screen for 2 seconds, as we can see in the following Figure 3.19.

Figure 3.20: Connection of addition with server

Part Three: In the first EditText, as shown from the hint of the EditText. The user of the application will enter a name. The second and third EditText is for entering two numbers for addition service, as shown in the following Figure 3.20:

Figure 3.21: Entering the application data

Part four: After entering the name and two numbers, those data will send to the server side application to use the addition service. Then, the server will run the addition function when the user clicks on the (Addition) button and send back the result of the addition to the client application. The result will be shown as a name followed by the result of the addition, as shown in the following Figure 3.21:

Figure 3.22: Client gets result back from server

3.2.2.3 The multiplication service implementation

Part one: Once the application is running on the client android machine, the welcome screen will be displayed, as shown in the following Figure 3.22, for 2 seconds.

Figure 3.23: Welcome screen of multiplication application

Part two: The following Figure 3.23, shows the second activity for the same client connecting to the server, as we can see in this Figure, by showing the message (Service Connected), the client user will be sure that the application is successfully running, and it is connected to the server:

Figure 3.24: Connection of the multiplication with server

Part Three: In the first EditText, as clear from the hint, the user of the application has to enter a name. In the second and third EditText will enter two numbers, as shown in the following Figure 3.24:

Figure 3.25: Entering the application data

Part four: After connection, the client with the server and after the data will be entered, by clicking the (Multiply) button the client gets the Multiplication process result, as shown in the following Figure 3.25:

Figure 3.26: Client gets the result back from the server

The Figure 3.25 shows the result of multiplying two integer numbers, which done by the server side. Then, the result is sent back to the client to show on the client side screen.

3.2.2.4 The myEncryption service implementation

This service is working as specific message encryption using some String methods and syntax in java programing language, it is necessary to mention that (myDecryption) does the decryption process for (myEncryption) service.

Part one: Once the application is running, the welcome screen will be displayed for 2 seconds, which is the first activity of the application, as shown in Figure 3.26:

Figure 3.27: Welcome screen of the myEncryption application

Part two: The following Figure 3.27, shows the second activity for the same client connecting to the server, as we can see in this Figure, by showing the message (Service Connected), the client will be sure that it is connected to the server with no errors:

Figure 3.28: Connection of the myEncryption with the server

Part Three: In the EditText, as clear from the hint the user of the application will enter a message, as shown in the following Figure 3.28:

Figure 3.29: Entering message for encryption

Part four: After connection, the client with the server and after entering the text message, the server will receive the entered information. The server will encrypt the entered message; then, the client will get the encryption process result, as shown in the following Figure 3.29:

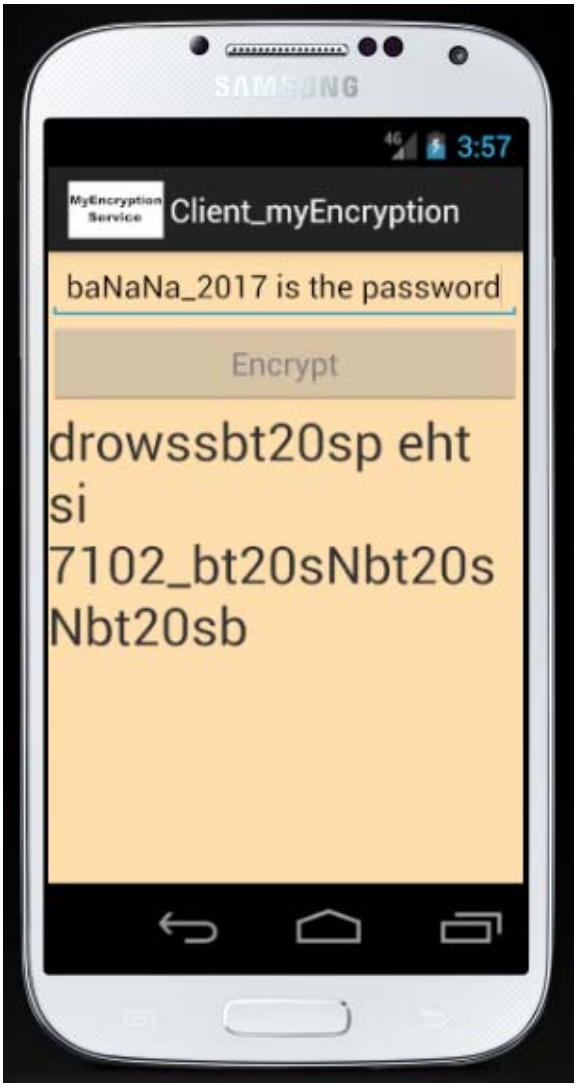

Figure 3.30: Client gets the result back from the server

3.2.2.5 The myDecryption service implementation

This service is working as a specific message decryption using some String methods and syntax in java programing language. Through this service, the client user can decrypt the encrypted message from the (myEncryption) service to get back the original message.

Part one: The Client which is request myDecryption service will be run on the same Android device. Once the application is running, the welcome screen will be displayed for 2 second, which is the first activity of the application, as shown in the following Figure 3.30:

Figure 3.31: Welcome screen of myDecryption client

Part Two: The following Figure 3.31, shows the second activity for the same client; which is connecting to the server, as we can see in this Figure, by showing the message (Service Connected), the client will be sure that it is successfully connected to the server with no problems:

Figure 3.32: Connection of the myDecryption with server

Part three: As shown in the following Figure 3.32, the user of the application has to enter the message; which encrypted by the myEncryption application for decryption.

Figure 3.33: Entering the encrypted message for decryption

Part Four: After the message will be entered, the client gets back the decryption process result, as shown in the following Figure 3.33:

Figure 3.34: Client gets the result back from the server

The Figure 3.34 shows the result of the myDecryption application.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Conclusion

This chapter includes two main parts. In the first part, the most important points of designing and implementing this work will be concluded. In the second part, the future works will be suggested. The mobile application is *becoming* advanced day by day, and becoming increasingly important part of peoples' daily lifestyle. Now days, people are able to use smartphones not only for calling and messaging service (SMS), but also for doing various computing tasks.

One of smartphone development challenges is related with the memory usage. Because they have a specific memory size compared with other devices such as laptops or desktop computers. So, running multiple applications on the same device will consume the memory and negatively affect the mobile performance; which makes software developers take into account develop applications with small memory usage.

One of the best techniques which help to decrease the memory usage is sharing services. The main idea of this thesis work is sharing services by designing and implementing a client server application; which are running on the same mobile device. The application has been developed for Android operating system using inter-process communication (IPC) and android interface definition language (AIDL).

The future works can be progressed to include many new parts using updated technologies and tools. The main future work can be summarized as the following:

- Developing and implementing this work for iOS devices, such as: iPhone and iPad.
- Developing the proposed system to support files and pictures, such as: encrypting files.
- Developing the proposed system to support faster data entry such as voice data entry.

REFERENCES

Android Application Components. Retrieved December 8, 2016, from https://www.tutorialspoint.com/android/android_application_components.htm.

Alina Gabaraeva, Presales Consultant Follow. (2015, March 14). THESIS_final_retouch-3. Retrieved April 13, 2017, from https://www.slideshare.net/AlinaGabaraeva/thesisfinal retouch3.

Android Developers. (2017). Services. Retrieved January, 19, 2017 from https://developer.android.com/guide/components/services.html

Android Developers. (n.d.). Retrieved December8, 2016, from https://www.bing.com/cr?IG=CE7E85548B93499B80863FE30BEF935E&CID=3A21FAC772B766501AA8F0A4732767F3&rd=1&h=M1RZlqxmw5Za8dNNGs0uOsABM3yF4A7N9DaqfoE_nU&v=1&r=https%3a%2f%2fdeveloper.android.com %2findex.html&p=DevEx,5063.1

Android Security: Guide to Android OS. (2016, October 27). Retrieved April 3, 2017, from https://www.veracode.com/security/android-security

Burd, B. (2012). *Android application development all-in-one for dummies*. Hoboken: John Wiley.

Burnette, E. (2015). *Hello, Android: introducing Google's mobile development platform*. Dallas, TX: The Pragmatic Bookshelf.

Career Guru. (2017). 50 Android Interview Questions & Answers. Retrieved from http://career.guru99.com/50-android-interview-questions-answers/

Darcey, L., & Conder, S. (2012). *Android Wireless Application Development Volume I: Android Essentials*. Addison-Wesley.

Encryption | Android Open Source Project. (n.d.). Retrieved April 3, 2017, from https://source.android.com/security/encryption/

Friesen, J. (2014). *Learn Java for Android development*. Berkeley, CA: Apress.

Griffiths, D., & Griffiths, D. (2015). *Head First Android Development*. " O'Reilly Media, Inc.". Haseman, C. (2012). *Creating Android applications: develop and design*. Berkeley, CA: Peachpit Press.

Gerri, O. (2017, January 19). Generating C Binder Interfaces with aidl-cpp. Retrieved June 26, 2017, from http://www.jianshu.com/p/5f2ae92291af

Holzner, S. (2004). *Eclipse cookbook*. " O'Reilly Media, Inc.". Implementing Remote Interface Using AIDL. (n.d.). Retrieved February2, 2017, from https://newcircle.com/s/post/48/implementing_remote_interface_using_aidl

Isakow, A., & Shi, H. (2008). Review of J2ME and J2MEbased Mobile Application. *International Journal of Communication and Network Security*, *8*(2), 189-198.

Jeong, K., Kang, H., Lee, K., & Park, S. (2015). Remote Binder: Remote Procedure Call between Android Devices. *KIISE Transactions on Computing Practices,21*(5),359-364.

Kris D. (2014, October 01). Introduction To Java – Learn Java for Android Development – Medium. Retrieved March5, 2017, from https://medium.com/learn-java-for-android-development/introduction-to-java-a95d84a681bd.

Lars Vogel. (2014). Android Services-Tutorial. Retrieved January, 22, 2017 from http://www.vogella.com/tutorials/AndroidServices/article.html#android-services.

Lee, W. M. (2011). Beginning Android Application Development. Indianapolis, Ind.: Wiley Publishing, Inc.

Liu, J. J. (2002). *Mobile map: A case study in the design & implementation of a mobile application* (Doctoral dissertation, Carleton University).

More than an Android emulator, a powerful virtualization platform to address all your Professional needs. (n.d.). Retrieved April 13, 2017, from https://www.genymotion .com/

Sbirlea, D., Burke, M. G., Guarnieri, S., Pistoia, M., & Sarkar, V. (2013). Automatic Detection of inter-application permission leaks in Android applications. *IBM Journal Research and Development, 57*(6). 10-1.

Singapati, S. (2012). Inter Process Communication in Android. *Master thesis, Tampere university of Technology*.

Schreiber, T. (2011, October). Android Binder–Android Interprocess Communication. In *Seminar thesis, Ruhr-Universität Bochum*.

Shah, S., & Rahman, K. A. (2013). *Android development tools for Eclipse: set up, build, And publish android projects quickly using Android development tools for Eclipse*. Birmingham:Packt Pub.

Research, C. S. (2012). Creative Android Apps. Retrieved February 14, 2017, from http://creativeandroidapps.blogspot.com/2012/06/introduction-to-android.html

Reto, M. (2009). Professional android application development. *Word Programmer to Programmer*, *6*(7), 1794-1797.

What is RSA algorithm (Rivest-Shamir-Adleman)? - Definition from WhatIs.com. (n.d.). Retrieved April 1, 2017, from http://searchsecurity.techtarget.com/definition/RSA