



Diploma thesis

# SmartCom

Secure SMS Encryption with  
High Usability

submitted by

Torsten W. Schröder  
student ID no. 1248471

Supervisors:

Prof. Dr. Joachim von zur Gathen  
Dipl.-Inf. Daniel Loebenberger

Department of Computer Security  
b-it - Bonn-Aachen International Center for Information Technology  
University of Bonn, GERMANY

March 12, 2012



# Declaration

I, Torsten W. Schröder (student of Computer Science at the University of Bonn, student ID no. 1248471), solemnly declare that I have written this thesis independently, and that I have not made use of any aids other than those acknowledged in this thesis. Neither this thesis, nor any similar work, have previously been submitted to any examination board.

Torsten W. Schröder  
Bonn, March 12, 2012

# Übersicht

Auf Grund der hohen Popularität und Nachfrage nach Smartphones wird diese Diplomarbeit einen Prototyp vorstellen, der die hohe Komplexität der Kryptographie mit einfacher Benutzbarkeit der Smartphones kombiniert. Das bekannte Android Betriebssystem, für das der Prototyp entwickelt wurde, bietet eine breite und umfangreiche Grundlage für die Entwicklung komplexer Applikationen.

Der Prototyp zeigt eine benutzeroptimierte Methode, um verschlüsselte Nachrichten zu senden. Während der Verschlüsselung wird eine Variante des Diffie-Hellman Schlüsselaustausches benutzt, die auf elliptischen Kurven beruht, um einen gemeinsamen Schlüssel zu generieren. Dabei wird auf eine SSL-Verbindung zurückgegriffen, die Zugriff auf einen Server ermöglicht, der die öffentlichen Schlüssel speichert. Im späteren Verlauf wird zum Speichern oder Senden der Nachrichten zu anderen Benutzern die AES Verschlüsselung verwendet. Der Vorteil dieser Kombination ist die Sicherheit gegenüber Man-in-the-Middle Attacks (basierend auf dem Design von SSL), gepaart mit den gründlich erforschten Grundlagen für den asymmetrischen Schlüsselaustausch und dem geringen zeitliche Aufwand des AES Algorithmus beim Verschlüsseln kurzer Nachrichten.

Ein anderer Aspekt der Implementierung ist die Einbettung des Programms in ein benutzerfreundliche, graphische Arbeitsoberfläche, dabei verwendet die Applikation speziellen Funktionalitäten der Smartphones und eine Verbindung zur Kontaktdatenbank.

## Abstract

Due to the popularity and the great demand for smartphones, this diploma thesis is going to introduce a prototype which establishes a connection between the complexity of cryptography and the simple usability of smartphones. The Android operating system, which was used, is widely known and offers a wide base for the development of complex applications.

The prototype shows a user-friendly method to send encrypted text messages. The encryption makes use of the elliptic curve Diffie-Hellman key exchange to create a common secret by using an SSL-connection to a key server. Later, the AES encryption is used for sending the text messages to other users and storing them in a database. The advantage of this combination is that the high resilience and reliability against man-in-the-middle attacks (due to the design of SSL) is paired with well-studied asymmetric primitives for key agreement and combined with the small time complexity of the AES for encrypting short messages.

Another aspect of implementation is the embedding of this program in a user-friendly graphical user interface and connecting it to special services and the contact-provider of the smartphone.

# Contents

|   |           |
|---|-----------|
| <b>Declaration</b>  | <b>i</b>  |
| <b>Abstract / Übersicht</b>                                   | <b>ii</b> |
| <b>1 Introduction</b>   | <b>4</b>  |
| <b>2 The Android Operating System</b>                         | <b>6</b>  |
| 2.1 Architecture of the Android-OS . . . . .                  | 7         |
| 2.2 The Dalvik Virtual Machine . . . . .                      | 8         |
| 2.3 The Android Sandbox Principle . . . . .                   | 9         |
| 2.4 Android integrated systems . . . . .                      | 10        |
| 2.4.1 The Contact Provider . . . . .                          | 10        |
| 2.4.2 The Notification Service . . . . .                      | 11        |
| 2.4.3 The Telephony Package . . . . .                         | 12        |
| 2.5 The Activity Lifecycle . . . . .                          | 12        |
| <b>3 Cryptography</b>   | <b>15</b> |
| 3.1 Elliptic curve cryptography . . . . .                     | 15        |
| 3.1.1 Elliptic curves . . . . .                               | 15        |
| 3.1.2 The Elliptic Curve Discrete Logarithm Problem . . . . . | 18        |
| 3.1.3 NIST curves . . . . .                                   | 18        |
| 3.2 Elliptic Curve Diffie-Hellman Key Exchange . . . . .      | 19        |
| 3.3 The SHA-256 Algorithm . . . . .                           | 21        |
| 3.3.1 Preparation . . . . .                                   | 21        |
| 3.3.2 The SHA-256 Message Schedule . . . . .                  | 21        |
| 3.3.3 The SHA-256 Compression Function . . . . .              | 22        |
| 3.4 The AES-Algorithm . . . . .                               | 23        |
| 3.4.1 Mode of operation . . . . .                             | 24        |
| 3.4.2 Key generation . . . . .                                | 25        |
| 3.4.3 Step: AddRoundKey . . . . .                             | 25        |
| 3.4.4 Step: SubBytes . . . . .                                | 26        |
| 3.4.5 Step: ShiftRows . . . . .                               | 26        |
| 3.4.6 Step: MixColumns . . . . .                              | 27        |

|          |  |           |
|----------|--|-----------|
| 3.4.7    | Decryption . . . . .   | 28        |
| 3.5      | The Secure Socket Layer . . . . .                              | 28        |
| 3.5.1    | The SSL Architecture . . . . .                                 | 29        |
| 3.5.2    | The SSL Handshake Protocol . . . . .                           | 30        |
| 3.5.3    | Additional first layer protocols . . . . .                     | 32        |
| 3.5.4    | The SSL Record Protocol . . . . .                              | 32        |
| <b>4</b> | <b>Design principles</b>                                       | <b>34</b> |
| 4.1      | How to create a good usability . . . . .                       | 34        |
| 4.2      | The golden ratio . . . . .                                     | 35        |
| 4.3      | Keep the interface transparent . . . . .                       | 36        |
| 4.3.1    | Think about the amount of elements on the screen . . . . .     | 36        |
| 4.3.2    | Present navigation differently . . . . .                       | 37        |
| 4.3.3    | Make use of established identifiers . . . . .                  | 37        |
| 4.3.4    | Group elements with the same theme . . . . .                   | 37        |
| 4.3.5    | Reduce distracting elements to a minimum . . . . .             | 37        |
| 4.4      | Minimize the text input . . . . .                              | 38        |
| 4.5      | Short ways for the user . . . . .                              | 38        |
| 4.6      | Take advantage of inbuilt functionality . . . . .              | 39        |
| 4.6.1    | Application menus . . . . .                                    | 39        |
| 4.6.2    | TabView and ListView . . . . .                                 | 40        |
| 4.6.3    | The AutoCompleteTextView . . . . .                             | 42        |
| <b>5</b> | <b>SmartCom - The Development</b>                              | <b>43</b> |
| 5.1      | Conflict between privacy and simplicity . . . . .              | 43        |
| 5.2      | Preparation . . . . .  | 44        |
| 5.3      | The development process . . . . .                              | 47        |
| 5.3.1    | The key exchange process . . . . .                             | 48        |
| 5.3.2    | The SmartCom key server . . . . .                              | 50        |
| 5.3.3    | Good usability by using specialized views . . . . .            | 50        |
| 5.3.4    | Support tools for improving the layout and usability . . . . . | 52        |
| 5.4      | Final processes . . . . .                                      | 53        |
| 5.4.1    | The login procedure . . . . .                                  | 53        |
| 5.4.2    | Storing and loading data . . . . .                             | 55        |
| 5.4.3    | The key exchange . . . . .                                     | 56        |
| 5.4.4    | Sending and receiving SMS . . . . .                            | 57        |
| 5.5      | Challenges during the development . . . . .                    | 58        |
| <b>6</b> | <b>SmartCOM - A demonstration</b>                              | <b>60</b> |
| 6.1      | Login . . . . .  | 60        |
| 6.2      | Key Exchange . . . . .   | 61        |

|          |                                  |           |
|----------|----------------------------------|-----------|
| 6.3      | Send SMS . . . . .               | 63        |
| 6.4      | Receive SMS . . . . .            | 64        |
| 6.5      | Last SMS . . . . .               | 65        |
| 6.6      | History . . . . .                | 66        |
| <b>7</b> | <b>Conclusion / Further Work</b> | <b>68</b> |
|          | <b>Bibliography</b>              | <b>73</b> |
|          | <b>Index</b>                     | <b>74</b> |

# Chapter 1

## Introduction

This diploma thesis is based on the diploma thesis of Thomas Berndt "CryptCOM - Insuring secure communication on arbitrary GSM phones by applying strong cryptography" [Ber10]. The author developed a prototype for SMS encryption for Java based mobile phones. This application offers a high level of security, but it is complicated to use as older generations of mobile phones have limited design features to support a good usability. Nowadays, the aspect of good usability becomes more and more important for application development. This diploma thesis will show a way to combine a high level of security and a good usability by using the basic ideas of Thomas Berndt combined with the new possibilities the smartphones offer.

Smartphones are the new generation of mobile phones. The first of these new devices, which was produced in great numbers and started the new hype, was the iPhone developed by Apple in 2007.

Compared to classic mobile phones, smartphones are optimized for many different applications. Older mobile phones are optimized for voice telephony and sending short messages. Apart from that, they are just able to run small java based applications. Only a few of these mobile phones are able to run more complex applications, but these applications are highly customized for individual phones or manufacturers and cannot be widely used.

Smartphones offer many complex applications in a large variety of categories. To support miscellaneous applications, a smartphone has many integrated subsystems. Therefore, it can be used as a communication center for telephony, handling SMS and MMS or as a personal information center to organize addresses, messages, personal notes and data like older mobile phones. These basic functions were extended by the possibility to organize e-mails and making video calls.

They can also be used as multimedia center to play music, show video clips and movies and for organizing this multimedia content. Additionally, they offer more



complex built in functionality for applications, for example an integrated GPS-sensor allowing the smartphone to be used as a satellite navigation system. Many sensors such as motion-, light- and approximation-sensors offer a lot of options for special applications, for example a spirit level or a distance meter. During the last years, these new electronical options became more and more popular in the game development for smartphones; at the University of Bonn, for instance, a research group has developed a variant of the "Scotland Yard" game for the iPhone, which uses the approximation sensor and the GPS for navigation and starting events [Bon].

In order to present complex applications, the smartphones do not only make use of their new, integrated systems but also utilize new possibilities for presentation and user interaction, like touchscreens and high-definition displays. Additionally, the new operating systems offer a large number of options to present information in a clear and concise way. Two well known and frequently used design instruments are the ListView and the TabView, see section 4.6.2.

The most important differentiating factor to classical mobile phones is the public and well described API, enabling the designer/programmer to create new applications and to extend existing applications with more ease. Normally, every operating-system for smartphones provides an option to include classes from established programming languages, the Android-OS can use java libraries which offers a wide scope for development.

## Chapter 2

# The Android Operating System

The new smartphones have many different operating systems (OS). Among these, around 85% market share fall to three well established OS for smartphones, the Apple iOS for iPhones ( $\approx 15\%$ ), Android ( $>50\%$ ) and Symbian OS ( $\approx 17\%$ ). The Android OS is becoming increasingly popular, with its market share rapidly growing [Gar10] [Gar11].

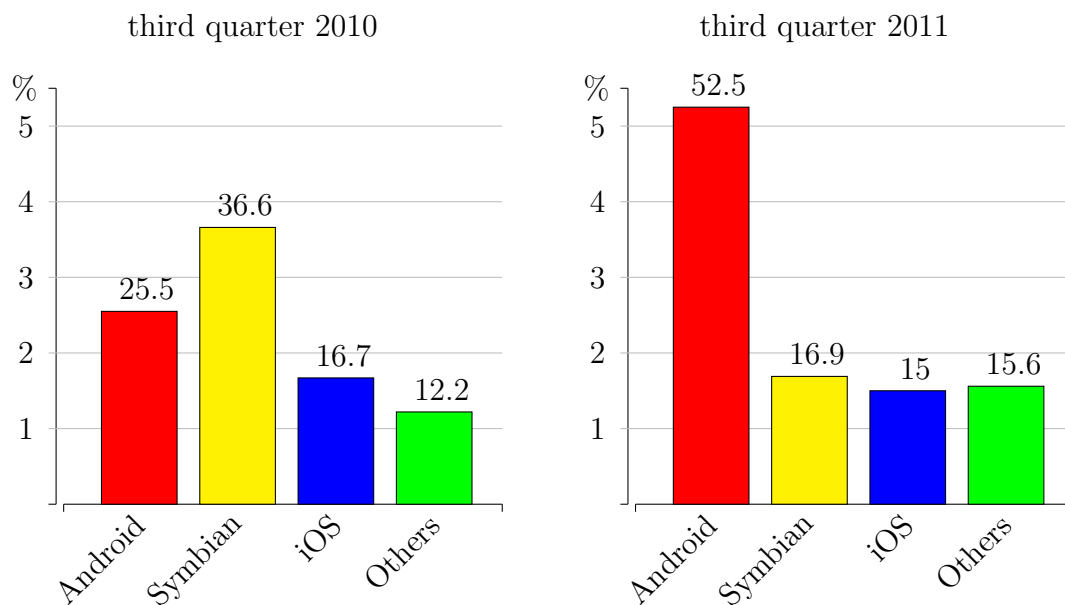


Figure 2.1: *Android market share following Gartner*

In the summer of 2005, Google bought the Android company which specialized in

location based applications for mobile phones. Since November 2007, the Android OS is developed by the "Open Handset Alliance" lead by Google. On October 21th of 2008, the first official version of the Android OS was made available [Bor08]. Initially, Android was designed for mobile devices like the smartphones, but nowadays it is used in several other fields because it is easy to install on devices with limited resources. Some producers of microprocessors have developed their own variant of the Android OS optimized for their processors, so that many embedded systems are running with the Android OS.

## 2.1 Architecture of the Android-OS

The content of this sections follows chapter one, section two of the book "Android 2 - Grundlagen der Programmierung" [BP10].

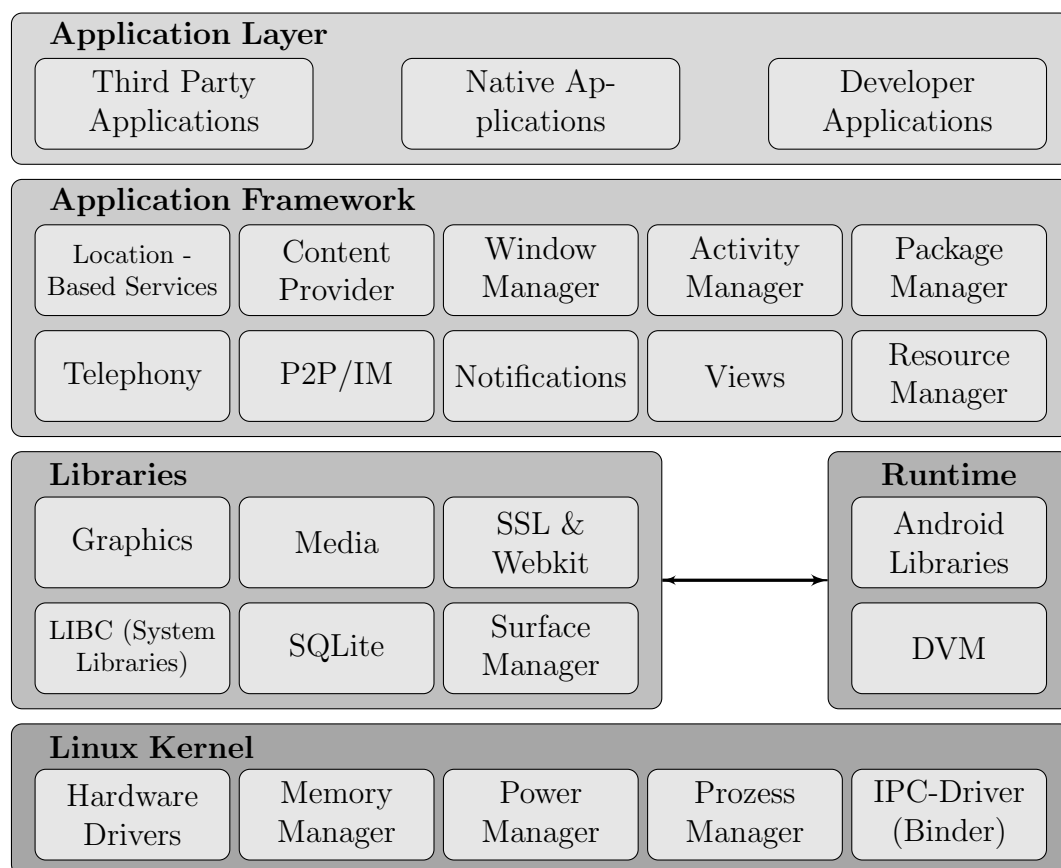


Figure 2.2: Android architecture

The Android OS has a four layer architecture, see Figure 2.2. The basic layer represents the Linux-Kernel and contains the hardware driver and controls the power, memory and process management. The kernel is optimized for a low energy and memory usage.

The second layer consists of the basic libraries and the runtime. The basic libraries are written in C/C++ and offer the core functionalities to run Android applications such as the database, a web-browser, the multimedia-management or the SSL components. The Android runtime contains the core components of the Android OS, the Android runtime libraries and the Dalvik Virtual Machine.

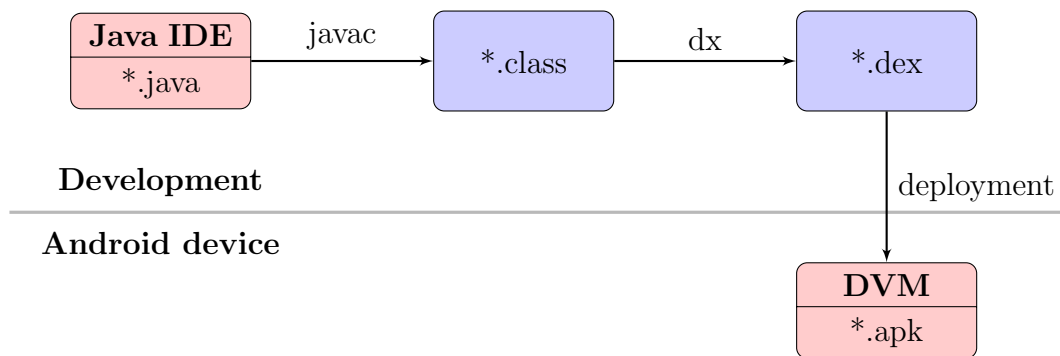
The next layer is the Application Framework layer. This framework allows access to system components by an application such as the contact database, basic phone information and status of services. In order to achieve this, this layer provides a wide range of manager classes which offer direct access to the different areas like the Content Provider (contact data), the Telephony Manager or the Window Manager. In contrast to the basic libraries, the manager classes are written in Java and form the foundation of application development.

The top layer is called the Application Layer. This layer contains the Android applications such as the standard applications developed by Google or third party applications. In this layer, the interaction between the user and the application takes place. In addition the communication between two applications is also regulated in the Application Layer.

## **2.2 The Dalvik Virtual Machine**

The Dalvik Virtual Machine (DVM) is the core component of the Android OS. It is based on the JVM Apache Harmony and was developed by Dan Bornstein, an employee of Google. The Apache Harmony JVM was optimized to keep the memory usage on a low level and to run multiple instances on a small device. In contrast to the JVM stack machine, the DVM was optimized for the modern processor architecture, so that the integrated registers of modern microprocessors can be used. Therefore, it has a register based architecture, which enables a faster calculation period for computations with many substeps.

At first, the DVM was developed and optimized for ARM-Processors used for embedded systems and mobile phones, because these processors are fast and have a low power usage. Nowadays, many manufacturers who have adapted this architecture, so that the Android OS can be used for several embedded systems, too.

Figure 2.3: *The Dalvik Virtual Machine*

Android applications are programmed in Java and later compiled to Java bytecode. After that, the Java bytecode is compiled to Dalvik bytecode by a program called DX (Figure 2.3). The DVM then uses the Dalvik bytecode to run the application.

## 2.3 The Android Sandbox Principle

Android runs its applications in a sandbox: a restricted runtime, in which it is not possible to directly access the OS and other applications running on this device. The Android OS is based on a Linux Kernel and every application follows the Linux principles. For this reason, every application has its own Linux-user and DVM, runs in its own process and has its dedicated section of the main memory and data storage. In addition, the Linux process- and access management is used, so that the directory in the data storage is not visible for other users and the process can not be accessed from outside without authorization. Therefore, no access to the DVM from the outside is possible.

Access to the application-process from the outside and accesses between application-processes can only be granted in the **AndroidManifest** [Andf]. The **AndroidManifest** is a XML file containing a detailed Activity (see section 2.5) description and the required rights of the application, for example the rights to send and receive SMS can be granted by adding the following code:

```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```

## 2.4 Android integrated systems

In comparison with older operating systems for mobile phones, the Android OS offers a number of providers and services to access the system data and functions. This is regulated in the android manifest as mentioned in the section above. In modern development, these providers and services are regularly used to provide more comfortable and user-optimized applications. Especially, many parameter entries can be omitted for the users, because these information can be read-out automatically.

### 2.4.1 The Contact Provider

The Contact Provider [Andnb] gives access to the contact data of the Android smartphone which contains definitions of supported URIs and columns.

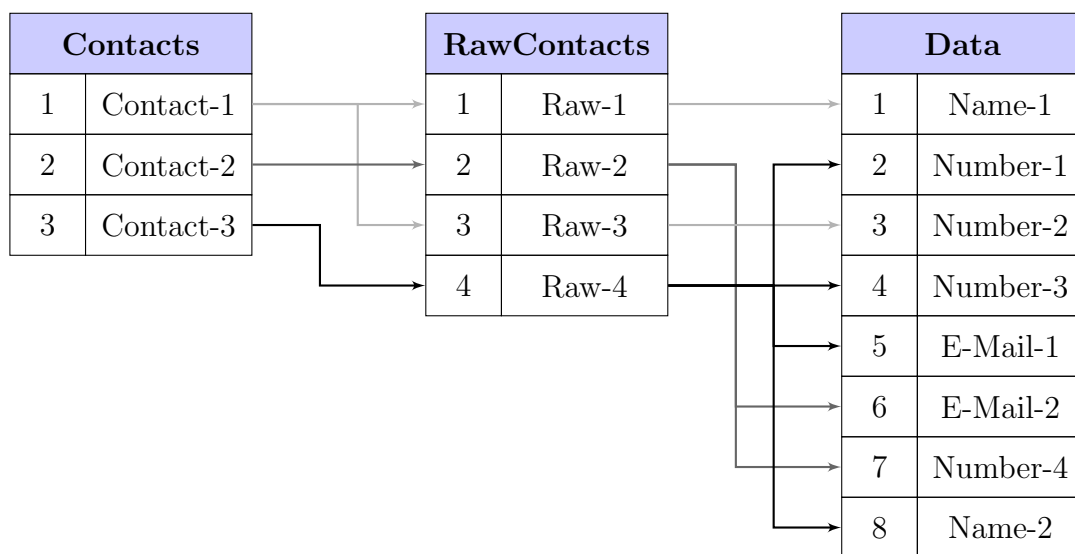


Figure 2.4: *Android: Contact Database*

In the contact database the data is stored in a three-tier model. The lowest tier consists of the `ContactsContract.Data` table. This table contains every kind of personal data, for example names, phone numbers or e-mail addresses. A predefined set of data types is available but every application can define new ones. The next

tier is composed by the `ContactsContract.RawContacts` table. A `RawContact` represents a set of data describing a person associated with a single account such as a GMail account. These `RawContacts` are merged into the top tier, containing the `ContactsContract.Contacts` table, so that every `RawContact` associated with the same person is represented by the same contact in the phone book. Figure 2.4 shows an example for three contact entries.

## 2.4.2 The Notification Service

The Notification Service [Andc] is used to alert the user about events happening in the background like receiving an SMS, a pending software update or a position update from the GPS. This service can be accessed through the Notification Manager. Notifications can have different forms:

- A persistent icon in the statusbar to start the corresponding application by clicking on this icon (Figure 2.5).
- The backlight or LEDs of the device can be turned on or flashed.
- A vibration can start.
- The user can be notified by a special sound.

Since applications became increasingly complex, many of them make use of services running in the background. Notifications are a simple and user-friendly way to keep the user informed about events which are important for their applications. In contrast to earlier OS, the user is always up to date even if the application is not running.

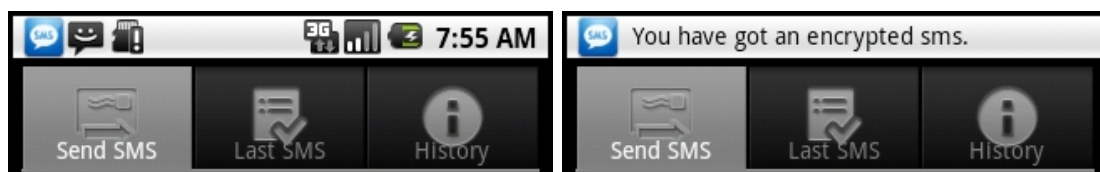


Figure 2.5: *Android: Notification (persistent icon)*

### 2.4.3 The Telephony Package

The Telephony Package [Ande] offers an API to get access to the basic phone information and functions. The package contains an SMS Manager, Telephony Manager and some utility classes for dealing with phone number strings. The SMS Manager is responsible for SMS operations like preparing and sending text or data SMS. The Telephony Manager establishes an interface to the Telephony service. This service offers access to the basic phone information such as location of the device, the signal strength or phone state. In addition, it allows the monitoring of the connection state and network type.

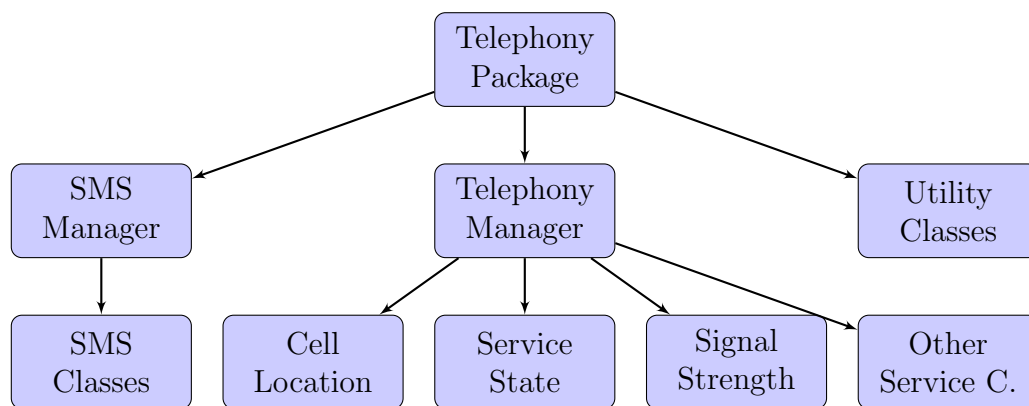


Figure 2.6: *Android: Telephony Package*

Applications make use of the Telephony Manager to register a listener to get notifications about state changes. The access to some telephony information requires permissions which can be declared in the `AndroidManifest` of the application. The simple access to basic phone information and functions offers a range of additional possibilities for application development in contrast to earlier OS.

## 2.5 The Activity Lifecycle

Another aspect of modern application development is the buffering of input information. The smartphone users want to have the option to switch between their applications whenever they want. In order to ensure that no data is lost, the developers have to think about mechanisms for buffering data and how to best integrate



these mechanisms into their program.

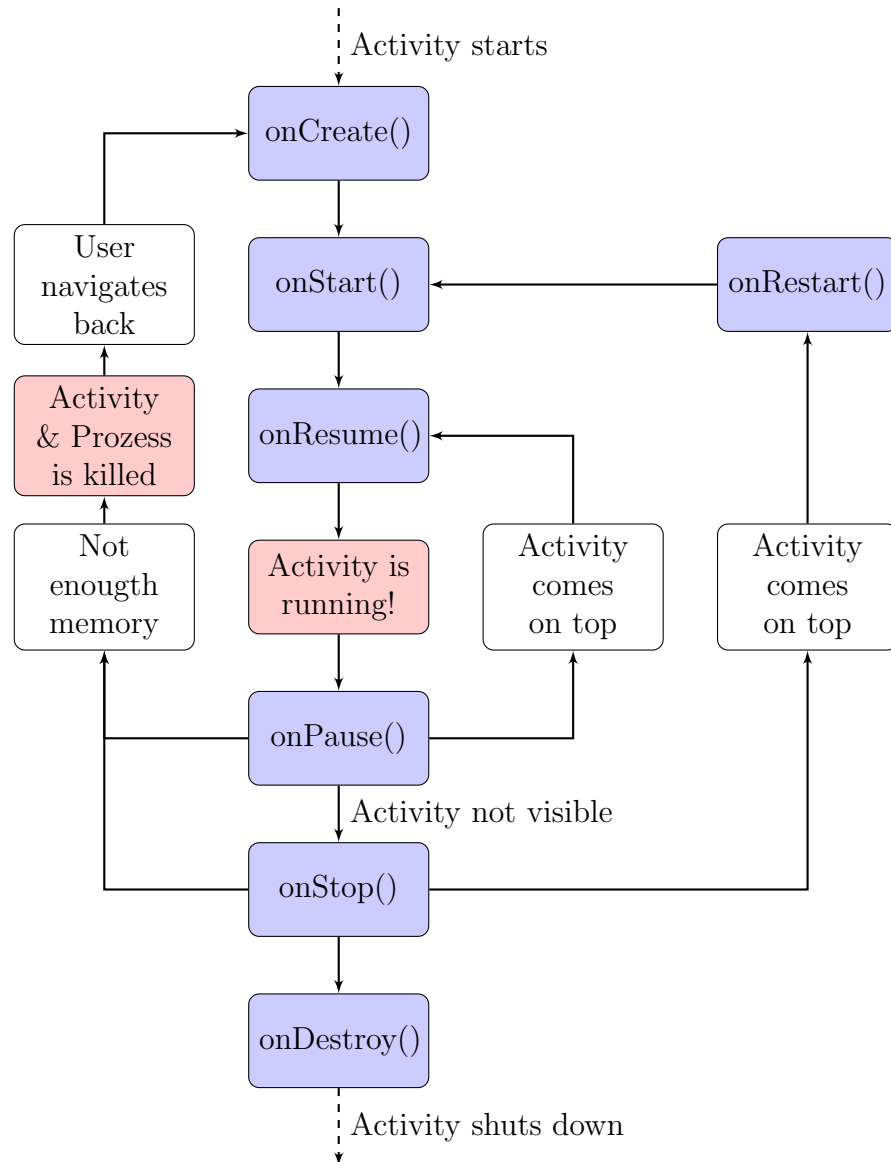


Figure 2.7: Android: Activity Lifecycle

The Android OS offers a special design of application which simplifies dealing with this aspect, called the Activity Lifecycle [Anda, BP10], see Figure 2.7. This cycle was developed to manage the memory and power usage since both resources are

very limited. As a side effect, it also makes it much easier to buffer inserted data. Every Application consists of one or more activities. An activity contains the view of an application, manages the view elements, handles the user input and responds to selections the user makes. The lifecycle describes the different states of an activity. Every activity runs through a predefined set of states; every state is started by calling the intended function of the state. These functions are also predefined, but can be adapted by overwriting or expanding them.

Android offers different integration points to deal with the buffering of data. The developers have many options to do so, depending on the special aspect they want to deal with, such as the handling of data filling a selection box. There are two simple ways to do that: On the one hand, if the input data changes rarely, one can load the data from the database in the `onCreate()` method, store it in a variable in the `onPause()` method and reload the variable and fill the selection box in the `onResume()` method. On the other hand, if the data changes rapidly, one can load the data from the database in every call of the `onResume()` method. The first implementation is the faster one but the second solution ensures that the information is always up to date.

# Chapter 3

## Cryptography

During the last few years, security has become ever more important for the users of personal computers and smartphones as many processes of their every-day life can be managed using the internet. Therefore, it is essential that the users can ensure their privacy in their banking transactions, mails and SMS. Key exchange mechanisms, authentication, efficient encryption algorithms and secure connections came more and more into the focus of application- and web- developers.

### 3.1 Elliptic curve cryptography

Elliptic curve cryptography (ECC) [Bun09] is one of the public key crypto systems. In a public key crypto system, every user taking part in the communication has a key pair: a public key and a private key. Additionally, each user must have a set of parameters for the encrypted communication, like the domain parameters, describing which elliptic curve is used for the calculation. The public key is not a secret, it is distributed to all users over an unsecured channel; in contrast, the private key is only known to the specific user. The asymmetric public key cryptography needs no common secret for communication but it is much slower than symmetric encryption.

#### 3.1.1 Elliptic curves

In order to understand the concept of ECC, this section will give the reader a short introduction into elliptic curves. An elliptic curve over a finite field  $K$  can

be defined as a set of points  $(x, y) \in K^2$  which satisfy the equation

$$y^2 = x^3 + ax + b \quad (3.1)$$

where  $a, b \in K$  are given.

The equation 3.1 is called Short Weierstrass Equation. In order to ensure that the elliptic curve is adapted for ECC, the parameters has to comply with the property

$$4a^3 + 27b^2 \neq 0. \quad (3.2)$$

This property ensures that the graph of the curve has no cusps or self-intersections. The points of the elliptic curve together with a special point of infinity  $O$  form an Abelian group, where  $O$  is the neutral element.

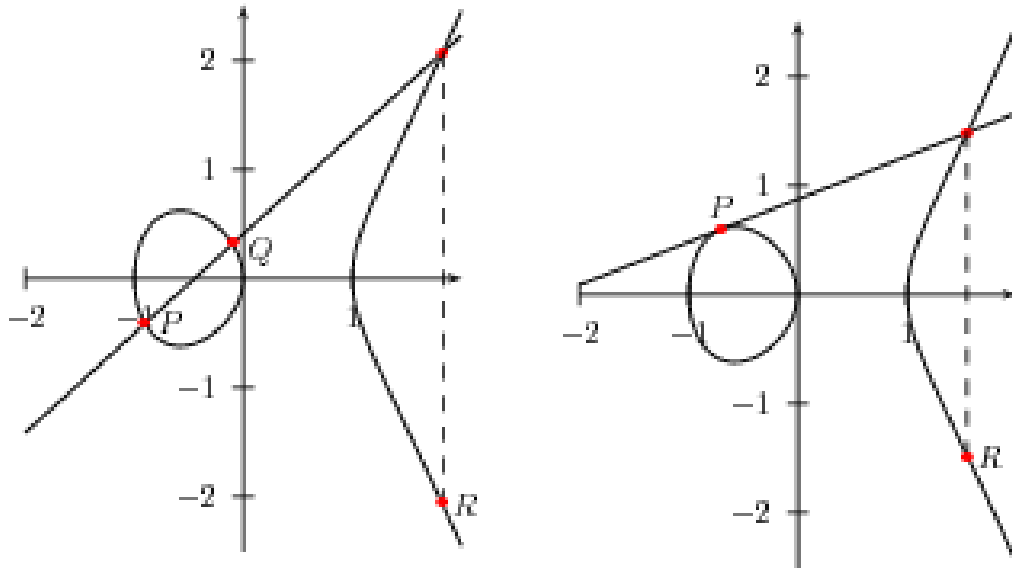


Figure 3.1: Addition and doubling on elliptic curves in  $\mathbb{R}^2$

In order to facilitate a better understanding, the concatenation for operating on an elliptic curve will be described for  $\mathbb{R}^2$ . Figure 3.1 shows in an illustrated way the two operations on an elliptic curve:

- **Adding two points** (on the left):

The line  $\overline{PQ}$  will cross the elliptic curve in a third point. Then the intersection point is reflected on the x-axis. The result of the reflection is also the result of the operation  $P + Q$ , the point  $R$ . If  $P$  and  $Q$  are lying on parallel line to the y-axis, the line will never again cross the elliptic curve and the result of the operation is the point of infinity  $O$ .

- **Doubling one points** (on the right):

The line  $\overline{PQ}$  is replaced by the tangent at the point  $P$ . The intersection point of the tangent and the elliptic curve is also reflected on the x-axis and the result is the result of the operation, the point  $R$ . If the tangent of  $P$  is a parallel line to the x-axis, the point of infinity  $O$  is the result of the operation.

After the illustrated description of the operations in  $\mathbb{R}^2$ , one can generally define the concatenation of two points  $P$  and  $Q$  over a finite field as follows:

$$P+Q = \begin{cases} P & , \text{ if } Q = O \\ Q & , \text{ if } P = O \\ O & , \text{ if } x_P = x_Q \text{ and } y_P \neq y_Q \\ (x_R, y_R) & , \text{ otherwise, with } x_R = m^2 - x_P - x_Q \text{ and } y_R = m(x_P - x_S) - y_P \end{cases}$$

Thereby the slope  $m$  can have two different values depending on  $P$  and  $Q$ :

$$m = \begin{cases} \frac{y_P - y_Q}{x_P - x_Q} & , \text{ if } P \neq Q \\ \frac{3x_P^2 + a}{2y_P} & , \text{ if } P = Q \end{cases}$$

The different cases for the slope  $m$  based on the two possible operations, described in an illustrated way before.

The algebraic rules for an elliptic curve in  $\mathbb{R}^2$  can be adopted for elliptic curves over a finite field  $\mathbb{F}_q$ , with the difference that the computations are performed in  $\mathbb{F}_q$ .

### 3.1.2 The Elliptic Curve Discrete Logarithm Problem

Asymmetric cryptographic mechanisms are typically based on some well-studied number-theoretic problems. A mechanism is strong, if this problem is difficult to solve in an efficient way. Former public key crypto systems such as RSA are based on the integer factorization problem, but nowadays algorithms were found to solve the integer factorization problem in subexponential time. As a consequence, RSA needs quite long keys for encryption to make it infeasible to attack. Newer public key systems like the ElGamal encryption are based on the Discrete Logarithm Problem (DLP). The ciphers using elliptic curves are based on the Elliptic Curve Discrete Logarithm Problem (ECDLP), which is similar to the DLP in the underlying finite field. Let  $E$  be an elliptic curve over the finite field  $\mathbb{F}_n$  and  $P, Q$  two points on the curve. Then the Elliptic Curve Discrete Logarithm Problem is defined as follows:

**Find  $n \in \mathbb{N}$  such that  $nP = Q$ .**

The ECDLP is believed to be much more difficult to solve in an efficient way than the DLP and in contrast to the integer factorization problem, there exists currently no known algorithm for solving the ECDLP in subexponential time.

This has an influence on the key size: much shorter keys can be used [LV00, NSA09a] to reach the same level of security as algorithms based on integer factorization or the DLP. For example, a 160-bit key in ECC is considered to be as secure as a 1024-bit key in RSA. Therefore, the ciphers based on the ECC are a favorite for the application development on mobile phones, where the resources are limited.

### 3.1.3 NIST curves

NIST curves are elliptic curves which are optimized for fast computations. It has been discovered that reduction modulo  $p$ , which is needed for addition and multiplication, is much faster if  $p \approx 2^d$ . This increase in speed has a practical background: Computers can do operations modulo  $p$  much more efficiently for numbers  $p$  near  $2^n$ , because they are operating on binary numbers including bitwise operations.

In FIPS 186-3 [NIS09] the NIST presents 15 elliptic curves over ten finite fields. Among these fields are five prime fields  $\mathbb{F}_p$  for certain primes  $p$  of sizes 192, 224, 256, 384, and 521-bit and five binary fields  $\mathbb{F}_{2^m}$  for  $m = \{163, 233, 283, 409, 571\}$ .

For each field an elliptic curve is recommended and in the binary case one Koblitz curve was also recommended. The curves have to provide optimal security and the implementation must be possible in a very efficient way, so that fast computations are ensured.

Even the National Security Agency (NSA) has included EC crypto systems working on these NIST curves into their Suite-B [NSA05] set of recommended algorithms. The Suite-B is used to protect classified information up to the "Secret" or "Top Secret" level.

The NIST is not the only institute which recommends curves in their standards, there are also two other established institutes, on the one hand the "American National Standards Institute" (ANSI) [ANS] and the "Standards for efficient cryptography group" (SECG) [SEC]. The prototype in this diploma thesis uses the "P-192" curve (NIST), this curve is one of two curves which are contained in all three standards [BWBG<sup>+</sup>06], but it can be replaced by every curve of the NIST standard, even by curves recommended for a 521-bit key size and thereby a level of security can be reached that is higher than the level recommended for "Top Secret" documents which uses a key size of 384-bit.

## 3.2 Elliptic Curve Diffie-Hellman Key Exchange

The Diffie-Hellman Key Exchange [DH06] is a cryptographic protocol which provides the possibility to exchange a common secret over an insecure channel. This secret is normally used to create a secret key for symmetric encryption algorithms like AES. The Elliptic Curve Diffie-Hellman Key Exchange (ECDH) [NSA09b][BJS07] is a variant of the original Diffie-Hellman key exchange which uses elliptic curve cryptography.

In order to use the key agreement protocol, both parties have to agree on the domain parameters, which form the basic structure the protocol is working on. These parameters are in the binary case  $(m, f(x), a, b, G, n, h)$  and in the prime case  $(p, a, b, n, h)$ . The first parameter defines a finite field, in the binary case  $\mathbb{F}_{2^m}$  by  $m$  and the irreducible polynomial  $f(x)$  over  $\mathbb{F}_2$  or in the prime case  $\mathbb{F}_p$  by the prime  $p$ . Then the elliptic curve  $E$  both parties have agreed on is described by the parameters  $a$  and  $b$  ( $y^2 = x^3 + ax + b$ ). The next parameter is the generator, a point  $G$  on the curve which defines a cyclic subgroup of  $E$ . The parameter  $n$  is the order of  $G$ , that means  $n$  is the smallest positive number solving the equation  $nG = O$  and it should contain at least one large prime divisor. The last parameter is the cofactor  $h = \frac{|E|}{n}$  which should preferably be near 1. These parameters can be passed over an insecure channel, none of them must be kept secret.

In the next step, the two parties, let me call them Alice and Bob, have to generate their private and public keys. The private key  $r$  is a randomly selected integer in the interval  $[1, n-1]$  and the public key  $PK = rG$ . Let  $(r_{Alice}, PK_{Alice})$  and  $(r_{Bob}, PK_{Bob})$  be the two key pairs. Now Alice and Bob exchange their public keys via the insecure channel.

Now Alice and Bob have to compute the secret point on the curve. Alice computes  $(x_k, y_k) = r_{Alice} PK_{Bob}$  and Bob  $(x_k, y_k) = r_{Bob} PK_{Alice}$ . Both will get the same point on the curve, because  $r_{Alice} PK_{Bob} = r_{Alice} r_{Bob} G = r_{Bob} r_{Alice} G = r_{Bob} PK_{Alice}$ . The x-coordinate  $x_k$  of the common point represents the shared secret by convention. Figure 3.2 shows the line of action in an illustrated way.

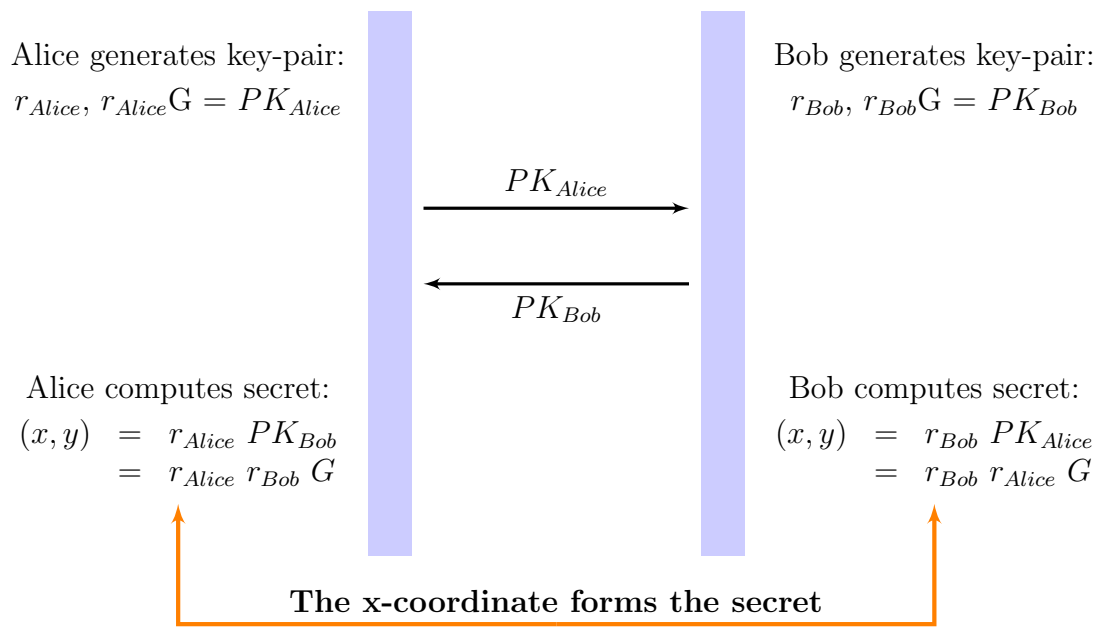


Figure 3.2: *Elliptic Curve Diffie-Hellman*

The ECDH protocol is secure unless a method is found to solve the Elliptic Curve Discrete Logarithm in an efficient way. The ECDH works perfectly, if the attacker can only read the information such as the domain parameters or the public keys, but if the attacker can alter some of the information, he can perform the agreement with both sides and thus observe the communication. This is commonly referred to as a Man-in-the-Middle attack. In order to avoid these Man-in-the-Middle attacks, the combination of Diffie-Hellman with an authentication mechanism is essential.



## 3.3 The SHA-256 Algorithm

The SHA-256 Algorithm [NIS02] is a cryptographic hash-function and belongs to the SHA2 family. It was developed by the National Security Agency and the National Institute of Standards and Technology. The hash-function was published in August 2002. The design of the SHA-256 hash function is based on the Merkle-Damgård construction and is intended to sign messages as part of the Digital Signature Algorithms (DSA) [NIS09]. The algorithm takes arbitrary blocks of data and returns a 256-bit string. The compression function of the SHA-256 operates on a 512-bit input and produces a 256-bit output.

### 3.3.1 Preparation

At first, the message is expanded by appending the "1" bit and as many "0"s so that the result is a multiple of 512-bit minus 64-bit. Then, the message length is added as a 64-bit word. Secondly, the message is split into 512-bit blocks. After that, every block is expanded a second time (this is called the SHA-256 Message Schedule) and the the algorithm is initialized with eight 32-bit words.

$$\begin{array}{lll} IV_1 = 6a09e667 & IV_4 = a54ff53a & IV_7 = 1f83d9ab \\ IV_2 = bb67ae85 & IV_5 = 510e527f & IV_8 = 5be0cd19 \\ IV_3 = 3c6ef372 & IV_6 = 9b05688c & \end{array}$$

### 3.3.2 The SHA-256 Message Schedule

During the SHA-256 Message Schedule every 512-bit block is split into 16 32-bit parts  $M_j$  and expanded to 64 32-bit words  $W_i$  which are needed for the 64 rounds of operations.

Operation:

$$\begin{array}{ll} \boxplus & = \text{Addition modulo } 2^{32} \\ \ggg & = \text{Rotation to the right} \\ \gg & = \text{Shift to the right} \end{array}$$

Support functions:

$$\begin{aligned} F_1(x) &= (x \ggg 7) \oplus (x \ggg 18) \oplus (x \gg 3) \\ F_2(x) &= (x \ggg 17) \oplus (x \ggg 19) \oplus (x \gg 10) \end{aligned}$$

Calculation of  $W_0, \dots, W_{63}$ :

$$W_i = \begin{cases} M_i, & \text{for } i = 0, \dots, 15 \\ F_2(W_{i-2}) \boxplus W_{i-7} \boxplus F_2(W_{i-15}) \boxplus W_{i-16}, & \text{for } i = 16, \dots, 63 \end{cases}$$

### 3.3.3 The SHA-256 Compression Function

After initialization and message expansion, the computation of the hash value begins. The algorithm runs through 64 equally designed rounds. In each round the next part of the message block  $W_i$  and the predefined round key  $K_i$  is added, every round has a different round key. In order to compute the result values of a round, four functions are used. All functions operate on a 32-bit word and produce a 32-bit word as output. The functions are defined as follows:

$$\begin{aligned} Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ Ma(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\ S_1(x) &= (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25) \\ S_2(x) &= (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22) \end{aligned}$$

During the calculation, the initial values  $IV_1, IV_2, IV_3, IV_5, IV_6, IV_7$  are shifted one to the right and the value for  $R_1$  and  $R_5$  is computed in the following way:

$$\begin{aligned} H_1 &= IV_8 \boxplus S_1(IV_1) \boxplus CH(IV_5, IV_6, IV_7) \boxplus K_i \boxplus W_i \\ H_2 &= S_2(IV_1) \boxplus Maj(IV_1, IV_2, IV_3) \\ R_1 &= H_1 \boxplus H_2 \\ R_5 &= IV_4 \boxplus H_1 \end{aligned}$$

At the end of each round, the result values  $R_1, R_2, \dots, R_8$  become the new initial values of the next round. The picture below describes one round of the SHA-256 algorithm in detail:

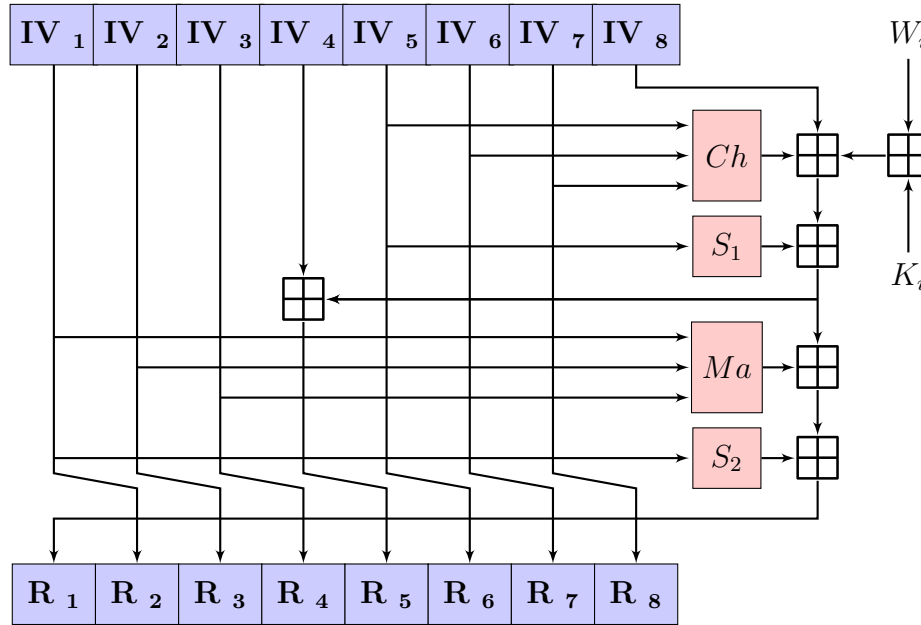


Figure 3.3: *SHA-256 Algorithm*

In the normal use of the SHA-256, one has many messages that are longer than 512-bit. In this case, the algorithm calculates one 512-bit block one after another and the hash value  $H$  is a combination of the hash values  $H_i$  before:

$$H_j = H_{j-1} \boxplus C_{M_j}(H_j - 1),$$

where  $C$  is the compression function of the SHA-256 algorithm.

## 3.4 The AES-Algorithm

The Advanced Encryption Standard (AES) is a specification for data encryption and was developed by Joan Daemen and Vincent Rijmen [DR02]. In October 2000,

it was announced as the successor of the Data Encryption Standard (DES) by the National Institute of Standards and Technology (NIST) [Bul00].

AES is a symmetric crypto system based on the Rijndael-Algorithm. In contrast to Rijndael, which can have a block- and a key-size of 128, 160, 192, 224, 256 bit, the AES algorithm has a 128-bit fixed block size and a variable key size of 128, 192 and 256 bit.

### 3.4.1 Mode of operation

AES operates on a  $4 \times 4$  matrix of bytes. The algorithm contains four operations distributed in rounds and the key expansion. At the beginning the key has to be expanded to get enough round-keys for encryption, because the original-key and also multiple parts of the expanded version are used.

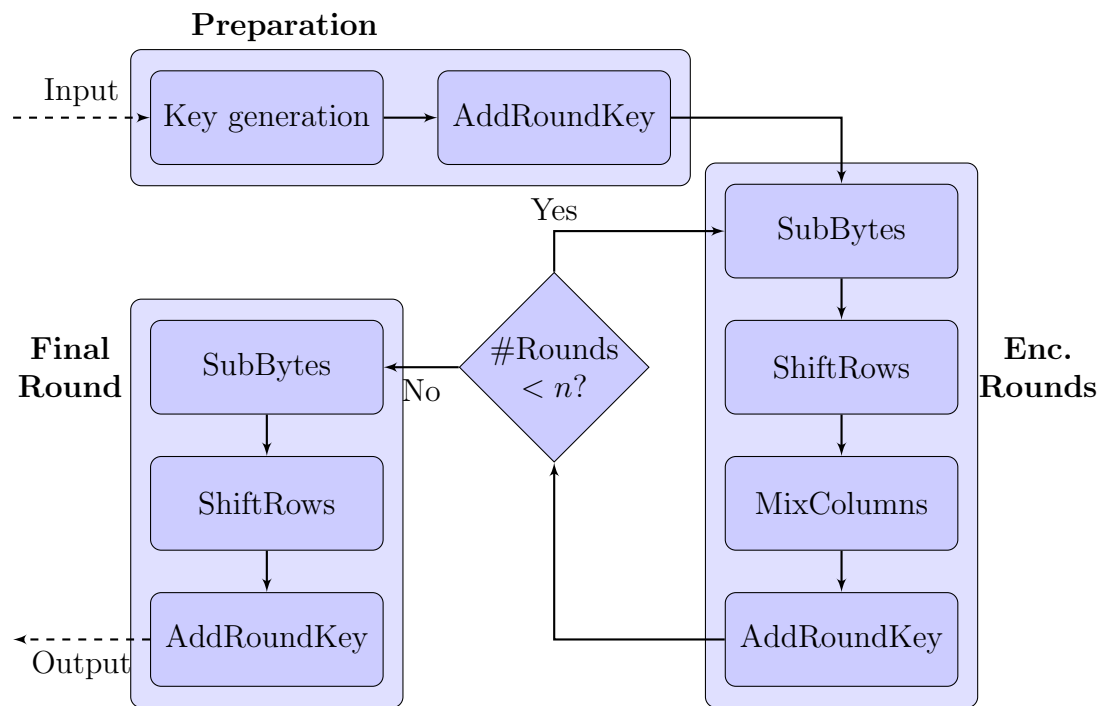


Figure 3.4: AES: Mode of operation

Then, every block of plaintext passes through four different preassigned transfor-

mations in a special order, the **SubBytes**, **ShiftRows**, **MixColumns** and **AddRoundKey** operation. The number of rounds which has to be passed depends on the selected key-size, 10 rounds for a 128-bit key, 12 for a 192-bit key and 14 rounds for a 256-bit key. Figure 3.4 shows a detailed overview of the operating sequence, where  $n$  is the number of the rounds.

### 3.4.2 Key generation

At first,  $(n + 1)$  round-keys have to be generated, where  $n$  is the number of rounds and the additional key is needed for the preparation round. The keys are derived from the secret key using the Rijndael key schedule.

### 3.4.3 Step: AddRoundKey

The **AddRoundKey** operation executes the key-addition. This means that every byte of the round-key is combined via the **XOR**-Function with a byte of the block containing the intermediate result of the encryption. This is the only step depending on the encryption-key.

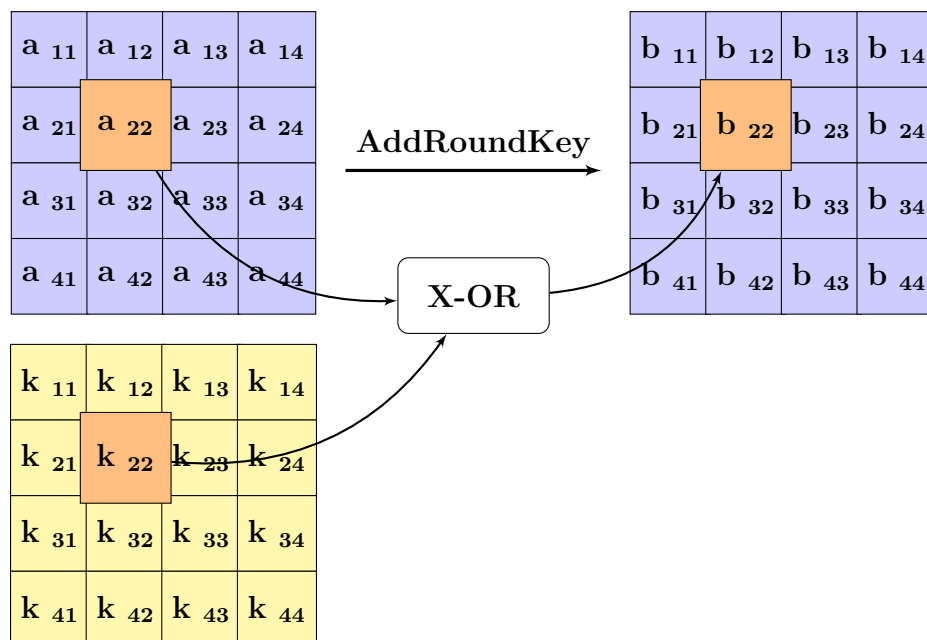
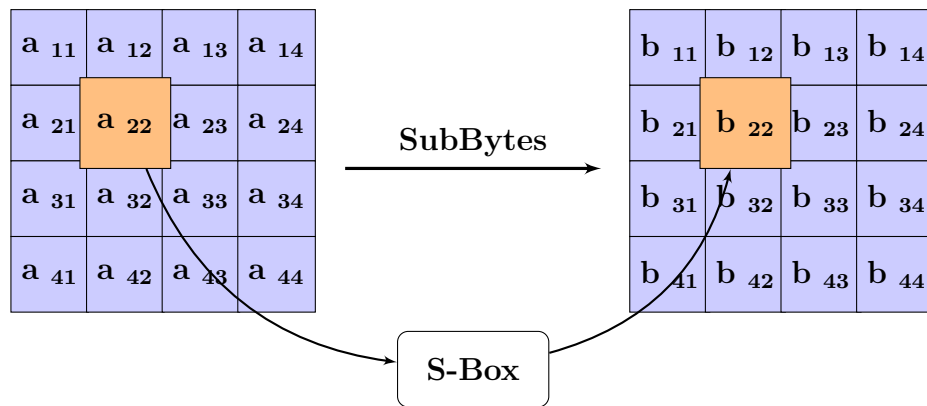


Figure 3.5: AES: AddRoundKey

### 3.4.4 Step: SubBytes

During the **SubBytes** operation, every byte of the block is transformed using an 8-bit substitution box, the S-Box or Rijndael-Box. This operation ensures the non-linearity of the encryption, because the S-Box based on the multiplicative inverse of  $\mathbb{F}_{2^m}$  which is known to have good non-linear properties. **SubBytes** has the functionality of a substitution cipher and as a result attacks depending on basic algebraic properties can be avoided.

Figure 3.6: AES: *SubBytes*

### 3.4.5 Step: ShiftRows

The execution of the **ShiftRows** operation is a cyclic left-shift of the bytes in the intermediate result. The second row is shifted once, the third twice and the fourth three times, only the first row will be left untouched. Every overflow on the left will be continued on the right side of the row. The **ShiftRows** step provides the diffusion of the columns.

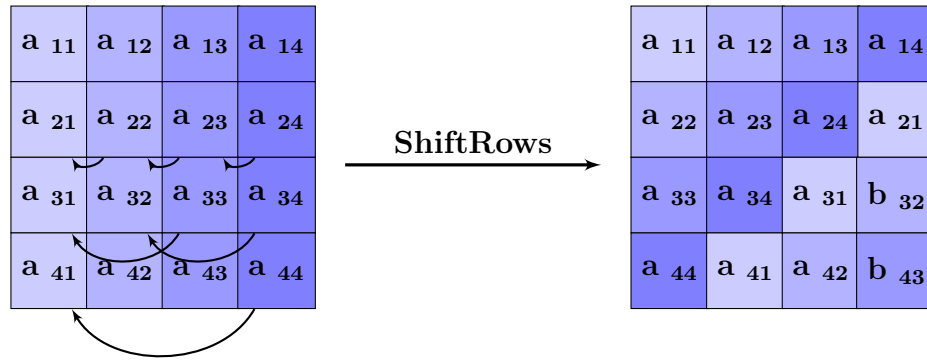


Figure 3.7: AES: *ShiftRows*

### 3.4.6 Step: MixColumns

In the **MixColumns** operation, every byte of a column is computed in a way that is affected by all the values in the column. Thereby, the four bytes of the column are combined by an invertible linear transformation. The transformation works on the Rijndael Galois field  $\mathbb{F}_{2^8}$ .

$$\begin{pmatrix} b_{1i} \\ b_{2i} \\ b_{3i} \\ b_{4i} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{1i} \\ a_{2i} \\ a_{3i} \\ a_{4i} \end{pmatrix}$$

The multiplication by 1 leaves the byte unchanged, multiplication by 2 is a shift to the left and the multiplication by 3 means left shift with the byte being combined by XOR with the initial value at this position. In mathematical terms, each byte of the column describes a polynomial over  $\mathbb{F}_{2^8}$ , this polynomial is multiplied modulo  $M(x) = x^4 + 1$  with the fixed polynomial  $a(x) := a_3x^3 + a_2x^2 + a_1x + a_0$  with the coefficients  $a_0 = 2$ ,  $a_1 = a_2 = 1$  and  $a_3 = 3$ . These coefficients were chosen to ensure the existence of the inverse elements, which is important for the decryption. The **MixColumns** step provides the diffusion of the rows. The operations **MixColumns** and **ShiftRows** ensure the diffusion of the cipher.

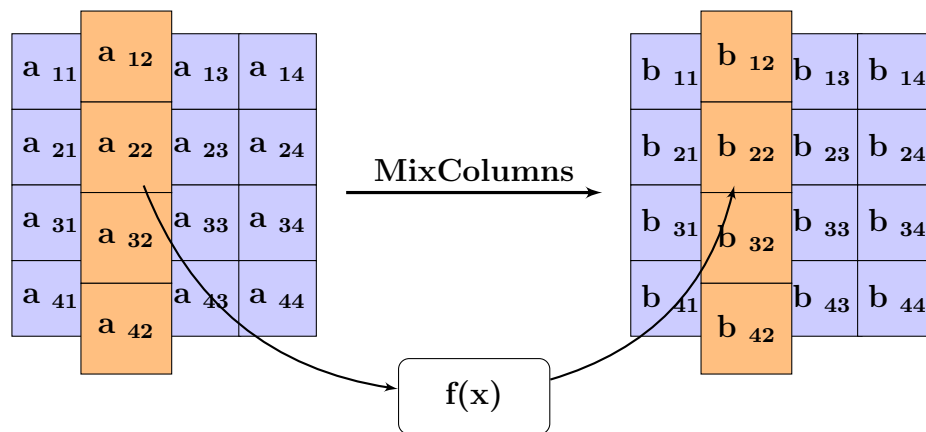


Figure 3.8: AES: MixColumns

### 3.4.7 Decryption

The decryption contains the same steps as the encryption but it is done in inverse order. During the **ShiftRows** operation the bytes are shifted in the other direction and in the **SubBytes** step an inverse S-Box is used.

## 3.5 The Secure Socket Layer

The Secure Sockets Layer (SSL) [FKK96, Andd] is an encryption protocol to ensure a secure data exchange over unsecure channels. Since Version 3.1, the Secure socket Layer is also known as Transport Layer Security (TLS) [DA99, DR08]. The protocol has two important functions:

- Create secure end-to-end connections.
- Authenticate servers and optionally the clients.

SSL is on the one hand a transparent security protocol, easy to use to create secure connections and on the other hand it has a flexible and forward-looking design, so that the encryption algorithms and the hash functions can be replaced by a newer version or by another algorithm or function.



### 3.5.1 The SSL Architecture

The SSL-Protocol combines different types of encryption mechanisms and offers a pool of encryption algorithms and hash-functions. Which set of algorithms is used later depends on the algorithms and key-sizes supported by the client. Generally, the SSL-Protocol uses:

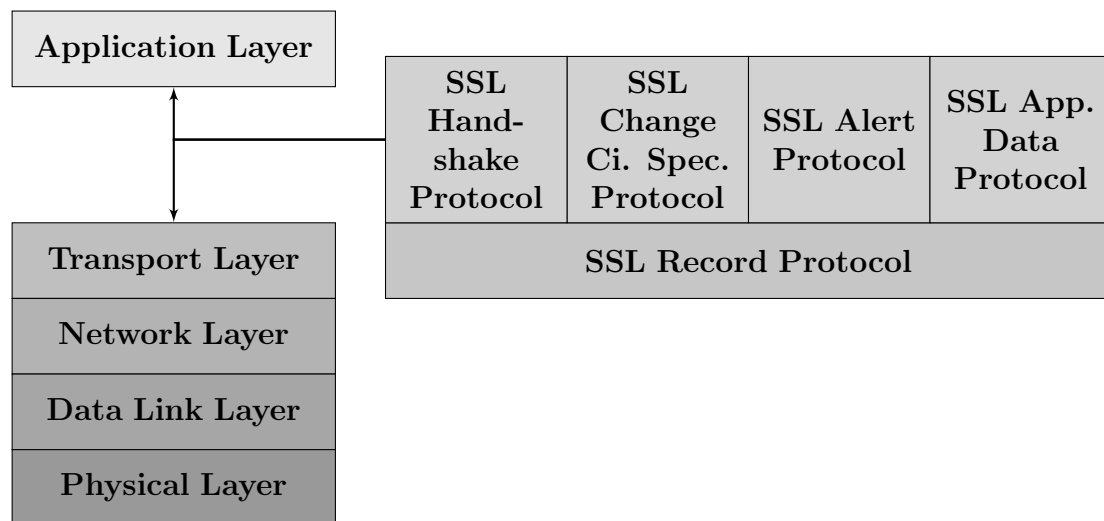
- Asymmetric encryption for the key-exchange (like RSA) and authentication.
- Symmetric encryption for the data-transfer for example AES or 3DES.
- Hash-functions like SHA256 to ensure the integrity of the data.

In the OSI-Layer-Model, SSL forms a new layer between the Transport-Layer and the Application-Layer [SG03]. Technically, the protocol establishes a secure connection between two sockets. A socket is a gateway to the TCP/IP-Network.

The function of the second layer is to start an authenticated, secure connection between client and server and to handle occurring warnings and errors. Therefore, an agreement on which set of algorithms and functions is used has to be reached and a common key has to be exchanged. The second layer is the basic layer, handles the data-transport and is responsible for the encryption and the integrity of the data. These two layers consist of five sub-protocols:

- SSL Record Protocol (forming the first layer)
- SSL Change Cipher Spec. Protocol
- SSL Alert Protocol
- SSL Application Data Protocol
- SSL Handshake Protocol

Figure 3.9 shows the position of SSL in the OSI-Layer-Model in an illustrated way.

Figure 3.9: *SSL: Protocol architecture and the position in the OSI-Model*

### 3.5.2 The SSL Handshake Protocol

The SSL Handshake Protocol is responsible for the authentication of the server and optionally of the client. The second important function is to reach an agreement with the server about the encryption algorithm which should be used for the encryption of the communication.

The protocol is executed at the beginning of an SSL session. It has a prescribed set of messages, which are sent between client and server. At first, the client asks for a secure socket by sending a plaintext message. This message includes the session ID, SSL version, 32-bit time-stamp, a 28-byte random number  $R_C$  and a list of the supported algorithms.

In a second step, the server sends back his certificate for authentication, the set of algorithms (cipher suite) which will be used and a time-stamp and random number  $R_S$ . Optionally, the server may ask for the client certificate. Many applications do not make use of the client-authentication as they use the password-authentication instead.

In the following step, the client has to verify the certificate. In case the certificate is valid, the client sends back his certificate and a signed hash over all previous messages for authentication (if required). Additionally, a new random  $R_{Sec}$  is generated and sent to the server encrypted with the public key of the server. Then the client computes the 48 byte Master-Secret based on the random numbers  $R_S$ ,  $R_C$ ,  $R_{Sec}$ . Following that, the keys for encryption are derived from the Master-Secret.

Finally, the client switches to an encrypted connection by starting the SSL Change Cipher Spec Protocol.

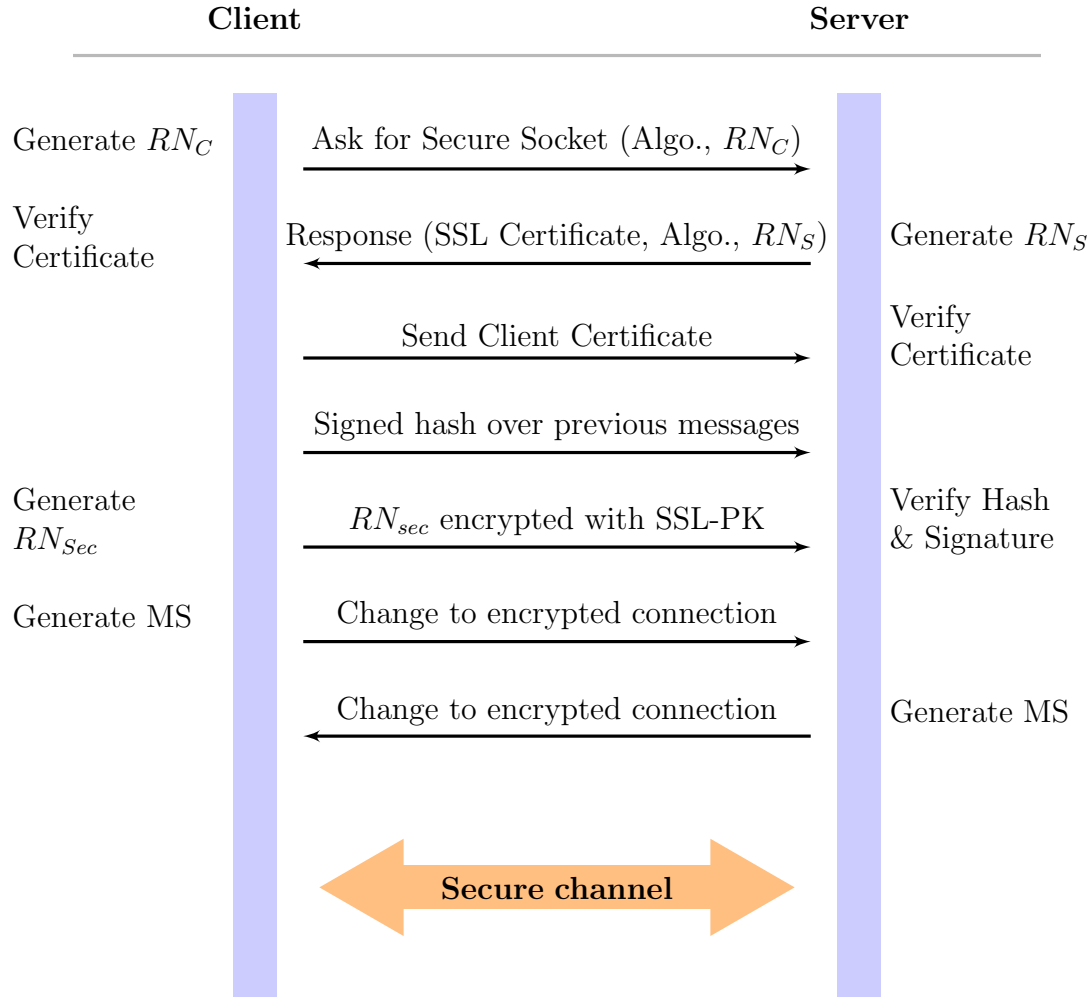


Figure 3.10: *SSL: Handshake Protocol*

As a last step, the server has to verify the client certificate and its validity. The random number  $R_{Sec}$  is then decrypted and the Master-Secret is calculated based on the random numbers  $R_S$ ,  $R_C$ ,  $R_{Sec}$ . The keys for encryption are derived from the Master-Secret and the SSL Change Cipher Spec Protocol is started. At that point a secure and authenticated connection is established. Figure 3.10 shows an illustrated description of the Handshake protocol.

Instead of the RSA variant, which uses the three random numbers for key creation, SSL also supports the Diffie-Hellman key exchange.

### 3.5.3 Additional first layer protocols

The Change Cipher Spec Protocol changes the encryption after new parameters have been exchanged. This happens by sending a single byte, which acts as a signal to switch to the new mode of encryption. It is also used in the SSL Handshake Protocol.

SSL Alert Protocol has to end the session by getting the "close\_notify" message and handles occurring warnings and errors. An error causes the immediate termination of the connection. Errors may be a handshake or an decompression failure. Warnings normally occur if one side has no or an invalid certificate or if there are problems with the decryption.

SSL Application Data Protocol manages the data transfer from the application to the Secure Socket Layer. It has access to the SSL Record Protocol.

### 3.5.4 The SSL Record Protocol

The SSL Record Protocol is responsible for the data transport. The protocol ensures the confidentiality and integrity of the data [Sta98].

Incoming data from the Application-Layer is prepared for the transfer. In that process the message is split into SSL-Records. Optionally, these records are compressed. Then the MAC (Message Authentication Code) is computed and added to the record, this ensures the integrity of the data. After that the package is encrypted. This should ensure the confidentiality of the data. Therefore, the algorithms and keys both sides have agreed on during execution of the SSL Handshake Protocol are used. Finally, the SSL-header is added. Figure 3.11 shows an illustrated description of the SSL Record Protocol:

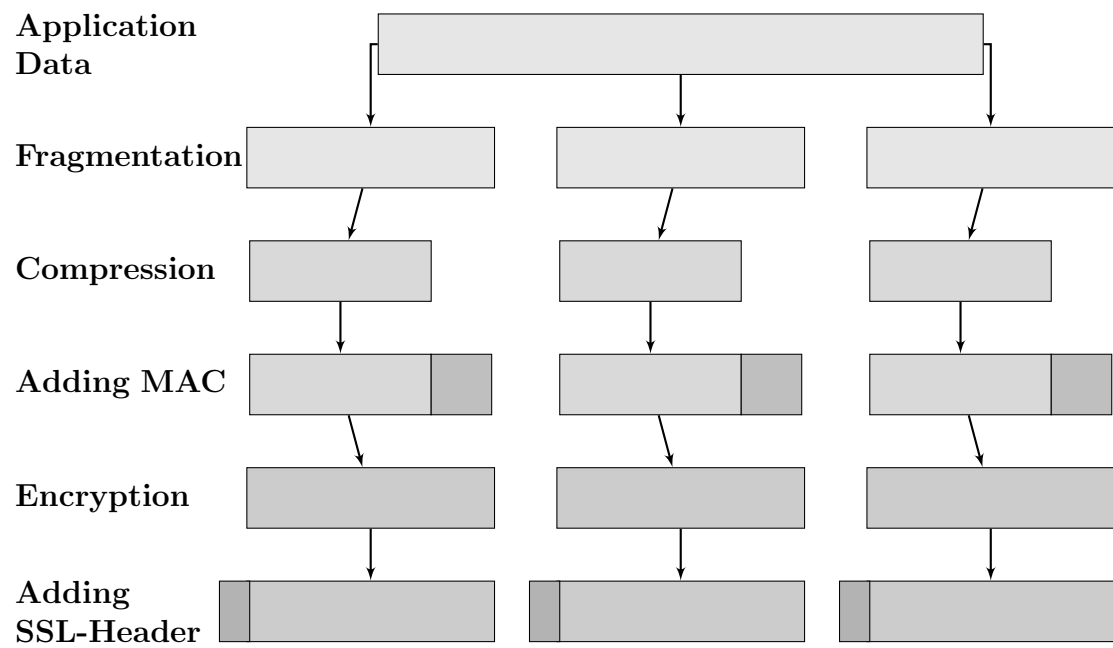


Figure 3.11: *SSL: Record Protocol*

# Chapter 4

## Design principles

Designing webpages or applications for smartphones is becoming more and more important as high-definition displays are supported and complex computations can be run in background. Information complexity increases and applications have to present the data in a clear and user-optimized way, or the users will ignore them. The developers of websites have long lasting experience creating good usability. Over the years, conventions were devised to create a standard for web development. Many of these design principles can be easily adapted for application development. This chapter will give a short overview of the important design principles.

### 4.1 How to create a good usability

The content of this chapter refers to the first chapter of the book "Modernes Webdesign - Grundgestaltungsprinzipien, Webstandards, Praxis" by Manuela Hoffmann [Hof10], where she describes design principles in a very general way.

In order to create good usability, two different aspects have to be kept in mind: the layout of an application and the technical utilities. Good layout ensures a transparent interface including a clearly arranged composition of the design elements, called "views" in Android. Android offers a wide range of special views to support or achieve a good design. In addition, the developer can make use of inbuilt technical features to support a good design, for example reading out addresses automatically from the contact database reduces the need to enter text manually.

### Important reasons for a good design:

- Follow the golden ratio to find the best position for the most important elements.
- Keep the interface transparent.
- Minimize text entry.
- Short ways for the user.
- Take advantage of inbuilt functionality.

## 4.2 The golden ratio

The golden ratio is a well-known natural law for proportions found in many parts of the modern world. It describes the proportion  $1 : \frac{1+\sqrt{5}}{2}$  ( $\approx 1,618$ ). Architecture, photography, or parts of the art have adapted the golden ratio for aesthetic reasons. In addition, the composition gains a natural order, harmony and stability.

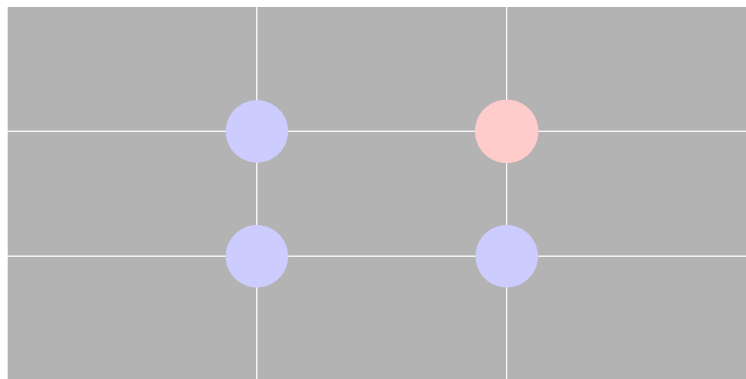


Figure 4.1: *The golden ratio in the simplified version (1 : 2)*

In web- or application development, a simplified version is commonly used, the proportion 1 : 2. Thereby, the interface is split into three horizontal and vertical fields. The most important element(s) should be placed at the intersection points of the stripline. The top right intersection point is thereby the point the user pays

attention to first. Figure 4.1 illustrates this simplified version with the top right intersection point highlighted in pink.

This method achieves a good result because the view elements of an application are typically larger and thereby a rough partition is adequate. In a more complex layout, it is necessary to use a more accurate proportion.

## 4.3 Keep the interface transparent

A transparent interface should give the impression of an appealing and well-arranged structure. The user should understand the interface not only by reading, but rather by seeing it, because shapes and images are noticed before any text is read. In order to keep the interface transparent, the developer/designer has multiple options:

- Think about the amount of elements on the screen.
- Present navigation differently.
- Make use of established identifiers.
- Group elements with the same theme.
- Reduce distracting elements to a minimum.

### 4.3.1 Think about the amount of elements on the screen

The structure of a webpage or an application is a key aspect of the development process. The designer has to think about the information or functionality he wants to present and prioritize the elements accordingly. Afterwards, he can create a basic structure focusing on the most important elements. In this process, the developer/designer has to find a balance between two aspects: On the one hand, the user's way to the desired functionality or information should be as short as possible (see section 4.5). On the other hand, that can result in more information per screen. The screen can quickly appear overloaded and overwhelming and the clear structure is lost. If this balance is disturbed, it is nearly impossible to create



an application with good usability, because all design possibilities discussed later in this section are based on the chosen structure.

### **4.3.2 Present navigation differently**

Navigation is very important to users, because they want to access the desired functionality or information as quickly as possible. Therefore, the navigation should be clearly visible, and the best way to achieve this, is to choose a special design for it. Normally the navigation menu is part of a bar, which is a shape the user easily recognizes and thereby avoids a time-consuming search. There are many ways to design the navigation, but it should have a distinct shape in order to be clearly recognizable.

### **4.3.3 Make use of established identifiers**

Established identifiers, like logos, make it much easier for the user to find his way to the desired information or functionality. The user recognizes pictures and shapes first, then the contained text. By using established identifiers, the user gets the information at first sight and doesn't have to read anything. It creates a clear and fast way and the user gets an appealing impression of the application.

### **4.3.4 Group elements with the same theme**

Grouping elements dealing with the same theme creates a structured interface because these shapes are also recognized first. In combination with established identifiers, the user quickly gets a good overview of the functionality and the possibilities the application offers.

### **4.3.5 Reduce distracting elements to a minimum**

What are distracting elements? The internet user knows webpage advertisements using the Adobe Flash-Player. These advertisements are colorful and gaudy design elements quickly getting the users attention. They are a good example for the

impression such elements make on the user. From a marketing point of view, it is a good idea to present these commercials in the foreground as the user will pay attention to them. But if there are too many of these elements, users will avoid the websites or applications because they are permanently distracted and kept from quickly reaching the functionality or information they seek. This aggressive advertising may easily overwhelm the user as it catches his eye before any other design elements and instead of instantly finding the right way to his destination, he has to take a second look in order to understand the interface. Even if the rest of the design is well thought-out, it is hard to change the negative impression if too many of these visually aggressive elements exist.

## **4.4 Minimize the text input**

Older applications sometimes require a long time for initialization and may have a somewhat confusing user guidance, because the user has to enter any required information manually. This results in applications containing many views for data input, making the application very complex and not clearly arranged. In addition, the ways for the user were very long to get to the required information or functionality.

Nowadays, every modern operating system for smartphones offers functionality to read-out data automatically. The Telephony package described in section 2.4.3 provides this functionality for Android smartphones, thereby reducing the described problems substantially. Evidently, minimizing text input has a large influence on the design of an application, although it may at first glance seem to be of minor importance.

## **4.5 Short ways for the user**

Short ways for the user are very important for good usability, because no user will enjoy an application if he has to spend a lot of time to reach the desired information or functionality and will quickly switch to another application.

Minimizing text entry is one possibility to create short ways for the user, but most important to achieving short ways is a well thought-out menu structure. Not all parts of an application must be accessible from every subpart. The developer and designer have to think about which connections are frequently used, rarely used

or even not used at all. In general, this is not easy, because in larger application the connections are more complex and every user has other priorities in the way he handles an application. The TabView offers a good tool for the implementation of an application which has only a few frequently used functionalities. These core functionalities can be accessed by only one click. A more detailed description of the Tabview will be given in the following section.

## 4.6 Take advantage of inbuilt functionality

The usage of inbuilt functionality is one very important aspect in the creation of a user-optimized interface, because these views or functions are adapted to the specific mobile-phone and can present the information in an even more optimized way or grant access to special information, for example the contact data of the user. The modern operating systems for smartphones offer a wide range of these design elements and functions. The Android-specific package, the Telephony package managing data access to special phone information, was described in section 2.4.3, in this section some Android-specific design elements, called "views", and the menus are described.

### 4.6.1 Application menus

In order to include functionality to change the look of the view or to add special functions, the Android system offers two types of menus: the option-menu and the context-menu.

The option-menu may be called by clicking the menu button of the smartphone; it creates a small box at the bottom of the page containing the buttons. This type of menu is good for including functions used rarely, because it interrupts the workflow, since the user has to change from the touchscreen to the menu bar of the smartphone.

The context-menu on the other hand can be called by a long click on the view element. The user stays near his point of action and can easily choose the required functionality. There is one more reason for the use of context-menu: the option-menu is only clearly arranged if there are not more than six functionalities in the background; the contextmenu has no such limitation, because the menu is scrollable and keeps its kind of presentation. Figure ?? shows a picture of an open option-menu on the left and context-menu on the right.

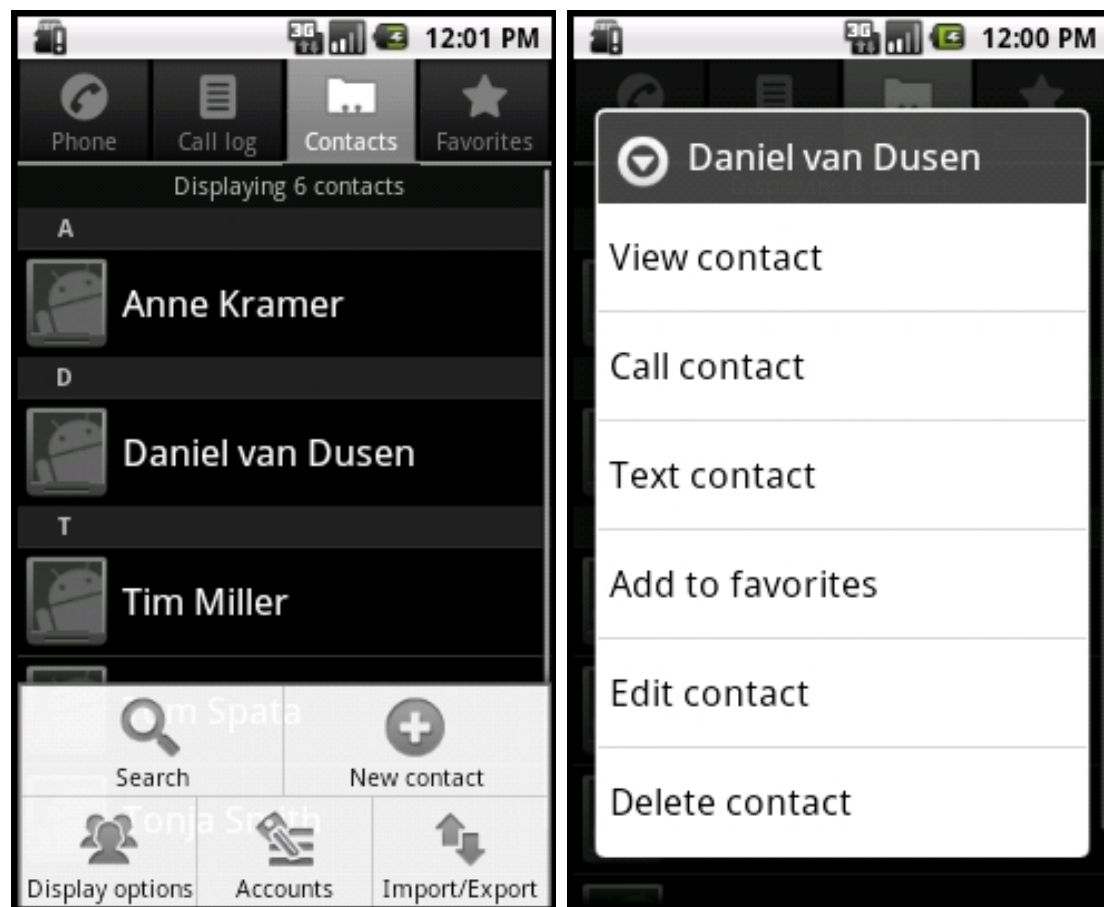


Figure 4.2: Layout: optionmenu and contextmenu

## 4.6.2 TabView and ListView

Two very popular views are the TabView and ListView. The TabView is especially suited to avoiding applications using many menus and submenus. The TabView splits the interface in two parts. At the top or the bottom (depending on the settings) a small button-bar is displayed. The rest of the interface can be used to show a view. This view can be easily changed by clicking or touching one button on the button-bar. It can be said that this view combines up to four views in one and provides a better level of usability, because it reduces the ways for the user and makes the interface clearly arranged and transparent. The use of a TabView with more than four buttons in the bar reduces the usability as the buttons become too small for interaction. Future generations of smartphones will have higher

resolutions, this limitation will be mitigated and the TabView will become even more attractive for developers.

The ListView is an optimized view for the presentation of a large amount of data. It is a scrollable view and can be connected to a database via an adapter. The presentation of a single list element can be changed in almost every way, every normal view element, even an image, can be used to design the list element. Thereby, a very clear and also user-optimized design can be created. Including graphics offers the possibility to integrate established identifiers for better ease of use and to reduce the amount of text in the view. Figure 4.3 show a simple TabView on the left and ListView on the right.

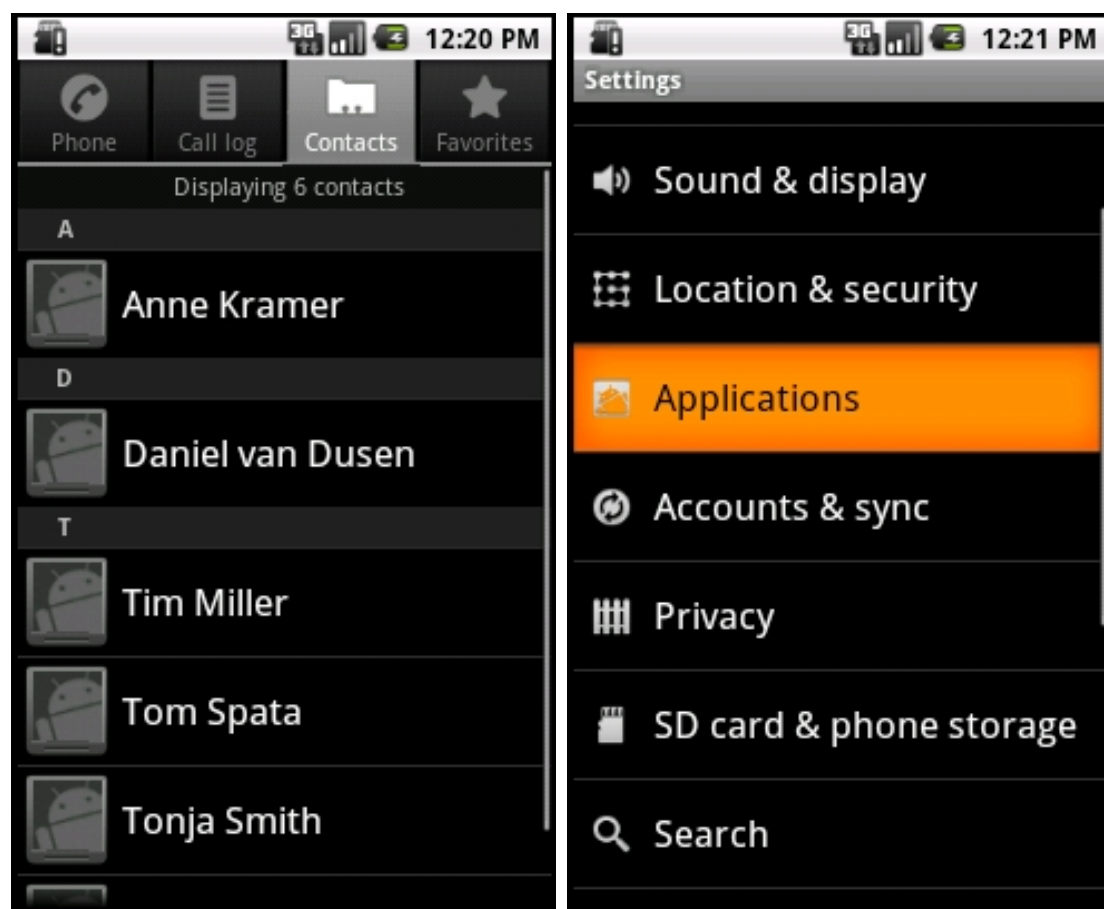


Figure 4.3: Layout: TabView and ListView

### 4.6.3 The AutoCompleteTextView

Another view which is used regularly is the `AutoCompleteTextView`. It can be connected via an adapter with a database or a predefined array to limit the possible input selection. This view looks like a normal text input field, but on entering a character, the appropriate preselection is shown below the text field. The selection is updated every time a change happens in the text field and the user can choose his favorite by clicking or selecting it using the touchscreen. The view offers a simple way to find specific information in a wide range of data. Figure 4.4 shows an `AutoCompleteTextView` with an open preselection on the left and a closed one on the right.

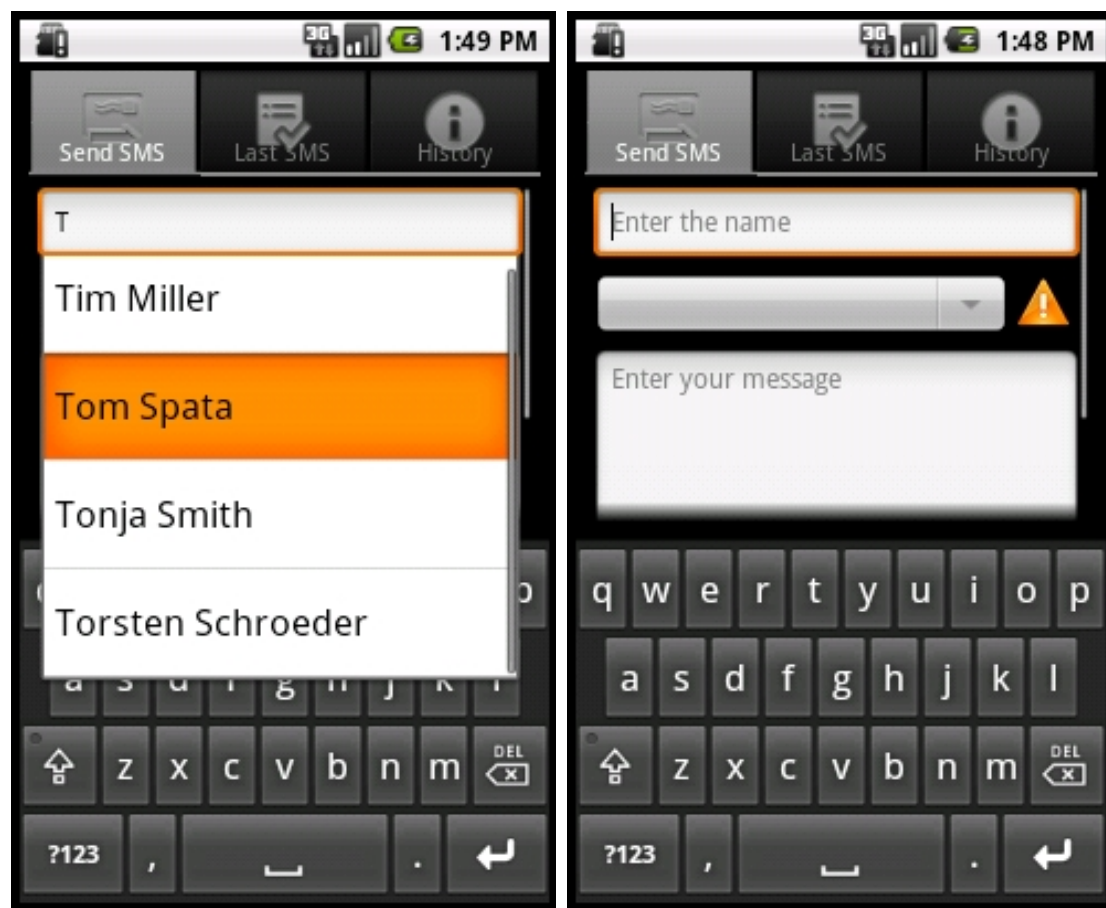


Figure 4.4: Layout: `AutoCompleteTextView`

# Chapter 5

## SmartCom - The Development

In this chapter, the development process of the SmartCom prototype is described; however, another factor has to be discussed beforehand: The conflict between privacy and simplicity. This debate has a big influence on the development process although it has nothing to do with design patterns or the layout of an application. This debate is essential to estimate the level of privacy a user wants to have. Developers need to balance two options: on the one hand they can demand a high level of private information to create an application that is very simple to use or on the other hand they can ask for almost no private information, but then the user has to do more by himself to initialize the application. Developers have to combine both options to create an optimized solution for their target group.

### 5.1 Conflict between privacy and simplicity

Today in particular, the conflict between usability and privacy concerns is more and more popular in the media as it is a global discussion. It has an impact on many aspects of digital life, like online shopping and payment, the usage of free-ware programs financed by advertising, and social networks. Many people will avoid applications which demand too much private information and also applications which are too complicated to use.

An increasing number of people are becoming aware of this conflict due to the public discussion about privacy on facebook and the user behavior. On facebook, every user can choose his level of privacy. In defining this level, the user can decide between strict privacy and a more powerful application, as high privacy settings will usually restrict the actions available to him. Since many people do not care a lot about the information they reveal, some of them make their whole life public

and do not think about the consequences.

In the internet community, two groups of people can be identified: The first consists of people, who like the easy way and are aware or do not care about the consequences ensuing from a lack of privacy. They want applications they can start and use quickly and do not wish to think about the handling. Therefore they reveal a lot about their person, because they allow the program to collect the information automatically. They only use fast and simple applications, even if another applications offers a much better solution for the problem. These users mainly think about how easy an application is to use and its speed. Waiting times will not be tolerated or accepted.

The second group of people attach great importance to their privacy and are reluctant to make even parts of their private life public. They care about security mechanisms and reflect the programs they use, even if they have to do more to initialize and configure programs to reach the same level of simplicity. If an application demands rights which can be used to collect private data automatically, the concerned user will often opt to stay away from such applications.

The Android OS offers a mechanism to control access rights. The developer has to declare the rights the application needs in the **AndroidManifest** (see section 2.3) and if the installation is started, the user has to accept these access rights, so that every user is informed about them and can estimate the consequences installing this program may have. This mechanism requires high individual responsibility as via the rights he is granting, it is possible to install malware or data mining software on his device.

During the next chapters, the development process of the SmartCom prototype is described, particularly with regard to the design decisions this user behavior results in.

## 5.2 Preparation

In order to work with both Android and a Server-Client architectures, it is important to have a development environment which contains customized tools and options for both. Eclipse offers a good solution:

- Eclipse Java EE environment for server side [Eclb].
- Eclipse extended with the ADT plugin for the client side [Ecla].



The Eclipse Java EE environment has an embedded web-server: the Apache Tomcat server [Apa]. The integrated development tools are optimized for programming servlets and handling errors like connection errors or failures occurring while handling a request or while creating a response for the client. The Eclipse platform needs no complex initialization; only for working with ssl-connections a ssl connector has to be added to the "server.xml" of the Tomcat server:

```
<Connector SSLEnabled="true"
  clientAuth="false"
  keystoreFile="D:/SmartCom/SmartComServer/keystore/.keystore"
  keystorePass="SmartCom"
  maxThreads="150" port="8443"
  scheme="https" secure="true" sslProtocol="TLS"/>
```

In this connector the keystore password can be changed by altering the code `keystorePass="SmartCom"`, but then the password of the keystore has to be updated, too. This can be done by using a program like "Protecle" [pro].

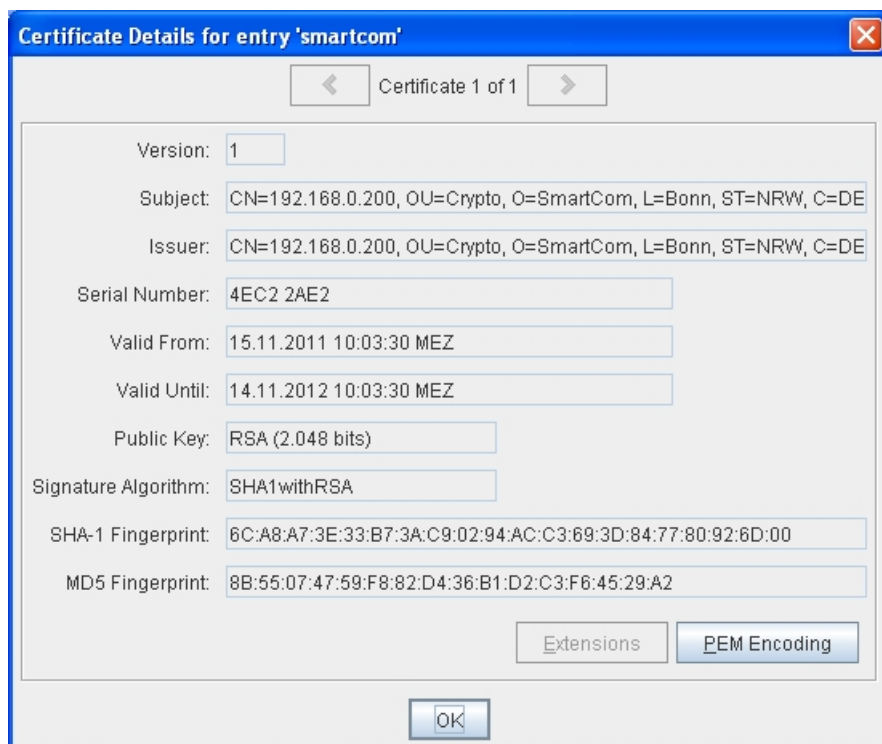


Figure 5.1: X.509 certificate

Additionally the X.509 certificate of the Tomcat server (see figure 5.1) has to be copied into the trusted keystore of the SmartCom application. The line of code `keystoreFile= "D:/SmartCom/SmartComServer/keystore/.keystore"` shows the path to integrated keystore of the Tomcat server. The trusted keystore of the SmartCom application can be found in the directory: `"D:/SmartCom/SmartComClient/res/raw/truststore.bks"`. If the project data was copied to another directory, the path has to be updated.



Figure 5.2: *Android Virtual Device*

Eclipse, extended with the ADT plugin, contains the Android libraries and has an integrated Android emulator tool, the AVD manager. The AVD manager enables the developer to work with multiple Android Virtual Devices (AVD) (see figure 5.2) and thereby allows to simulate more complex applications based on a server-client architecture.

In addition to the standard tools for debugging java code such as the analysis of data at a specific break-point, the environment has a special program for Android development. The LogCat is a custom tool for logging and displaying the standard Android or self defined entries while running the application on the emulator. This is very similar to log4j for java programs. The Dalvik Debug Monitor Server

(DDMS) gives a deep insight into the emulator during the test. The DDMS can, for example, show the heap, running threads or the file explorer. In addition it also facilitates working with SD Cards on the simulated device in an effective way.

## 5.3 The development process

After familiarization with the Android Framework to figure out the new possibilities for application development, especially the new design components and the usage of cryptography, I found out that only an older version of Bouncy Castle [Bou] crypto package is available in the standard Android libraries. Therefore, I decided to use the current version. Initially, this led to a problem, because in the Android Framework an older class is not overwritten if it is part of the Android libraries and that results in obscure errors. After recompiling the classes using another name, "bc2", the package could be imported the normal way.

The Bouncy Castle crypto package is a powerful library and has a wide range of supported cryptographic algorithms including algorithms for EC cryptography. It contains a light-weight API for use in any environment, including the newly released J2ME. The developers of the library attach great value to working with the newest specifications. The interested reader will find more detailed information on the website: <http://www.bouncycastle.org/specifications.html>.

During previous works, I realized that it is easy to deal with prototypes while working on a application, because it is easier to detect weak spots in the implementation and to get a feedback from users. This is very important in order to find a general solution accepted by most of the potential users.

The first prototype has almost the same functionality as the prototype by Thomas Berndt [Ber10] containing the Diffie-Hellman key-exchange and the authentication mechanism. The application was able to send SMS encrypted with AES, but it was on the same level of usability and therefore complex to handle. At the outset, I started to figure out what was responsible for the lack of usability. After some discussions with people who frequently use their smartphones for business, two main reasons became apparent:

1. The key exchange process takes too much time and cost money.
2. Due to the limited possibilities offered by older generation mobile phones:
  - The ways for the user are too long.

- No support tools are used to improve the usability.
- No specialized views are used for presentation.

### 5.3.1 The key exchange process

At first, solutions for a faster key exchange process had to be found since this is the main disadvantage due to which users prefer other solutions for SMS encryption. Android offers an additional possibility to develop a solution, the internet flatrate which normally every smartphone user has. Without a data flatrate, I found no better ways of dealing with the problem, but when thinking about solutions using a fast and readily available internet connection, two additional approaches to addressing the problem become feasible:

- Making use of official public keys from well-known public key servers.
- Using a special key server for SmartCom and generating new keys for every user at the first start of the application.

The use of the official public keys seems to be the easiest solution for the problem. However, the user has to reveal some private information. After long discussions with different people I chose the second option, for two reasons: First, many people who have a public key on a server were not willing to reveal their key for any application, even if it is only a public key which contains no secret information. Secondly, several people found it very complicated and time-consuming to create their own key only for using a special application. This discussion is a good example for the conflict previously described and the decisions a developer has to make to find a balance between the two groups.

In order to work with a key server which is not from a trusted key authority, the authentication has to be managed, because Diffie-Hellman is only secure if the message cannot be changed during the transfer. Otherwise, an attacker can use a Man-in-the-Middle attack to draw conclusions about the key used for the SMS encryption (see section 3.2). The SSL connection offers a mechanism to create an authenticated connection for both sides, but there are a few points to consider: If a bidirectional authenticated connection should be used, both sides need certificates for the connection. Every server normally has such a certificate, but the question remains what certificate should be used for the client side. Is it better to use a solution which combines only a server-sided authenticated connection with

another mechanism for the client authentication? After some research, I found three possible solutions:

- The client uses the certificate from a trusted key authority like Thawte [Tha].
- The server sends back a self-created certificate to the client for later communication.
- Only a server-sided authentication with the client using his mobile number for the authentication.

The first option seems to be the best solution for the problem, because it is easy to get the public keys from the server and it ensures an authenticated connection. Nevertheless in the background we have to deal with the same problem as in the discussion about the public keys: The user has to reveal his certificate to use the application and normally he does not know what is running in the background and for this reason many users refuse this option. If the user has no certificate from a trusted key authority, he has to register to get the certificate and to trust a third "party". In addition, the user has to pay for the certificate and that is the most important point, because only a few people have such certificates or are willing to spend money to obtain them.

The next step in the development process was to implement and test the second version. After some tests and longer discussions, it became clear that this would be a user-optimized solution, but it takes too much time from the perspective of the test subjects. The creation of the certificate and later the storing of the certificate takes no longer than ten seconds on the emulator, but if a person makes full use of the application, it is a tedious and time-consuming act. Especially people with many contacts, who want everything encrypted, will not use this application. This can be mitigated in the future as smartphones become more and more powerful.

The third version of the program can be found in web-applications, for example online-banking, but in this case a password is used instead of the phone number. The Telephony Package offers a method to read-out the phone data automatically (see section 2.4.3). As a result, the implementation of the third version is as user-friendly as the second solution but the interactions are much faster. The user gets a good start with the application, because the key exchange is the first step and many users think that fast processes are an important basic quality for an application with high usability (see section 4.5). The new key exchange process is described in detail in the following chapter.

### 5.3.2 The SmartCom key server

Due to the new version of the key exchange process, the SmartCom application makes use of a dedicated public key server. The server has a straight structure: In order to use an SSL connection, the server contains a keystore with trusted certificates. A servlet handles the requests of the clients and creates an individualized response for each request containing the requested key or a predefined message which informs the client that at the moment no key is available for the number contained in the query. Additionally, the servlet has a connection to the database to manage the incoming public keys. Every client sends its public key with the first request to the server. After that, other clients can ask for the key to create a common secret key for encryption.

### 5.3.3 Good usability by using specialized views

After the technical adaptations, I had to think about how to integrate the new possibilities the Android smartphones offer to create a clear and well-arranged application combined with short ways for the user (see chapter 4). I tested many entirely different solutions until I came to the final one. Therefore, I will only describe the underlying concept of the final version of the SmartCom layout. During the optimization process, I discussed every solution with people who use the smartphone very frequently, to get solutions for wide range of potential customers. The final version is based on a TabView layout (see section 4.6.2). This layout ensures fast access to all three parts of the application, the **Send SMS** part, the **Last SMS** and the **History** part containing a list of all received or sent SMS. After the start of the application the **Send SMS** part is in front by default. The **Send SMS** part is the most commonly used component of the application and thus has to be reachable quickly (see section 4.5). Many users like the short overview about sent and received SMS. With this layout, the user only needs to click once on the **Last SMS** tab to get this overview. In this summary only recent SMS are shown. If the user wants to find older ones from a special contact, the **History** tab offers this option containing a contact overview. After choosing the contact, a summary of all received and sent SMS becomes visible and the user can select a message to receive more detailed information about it. Figure 5.3 shows two examples for the TabView layout used in the SmartCom application.

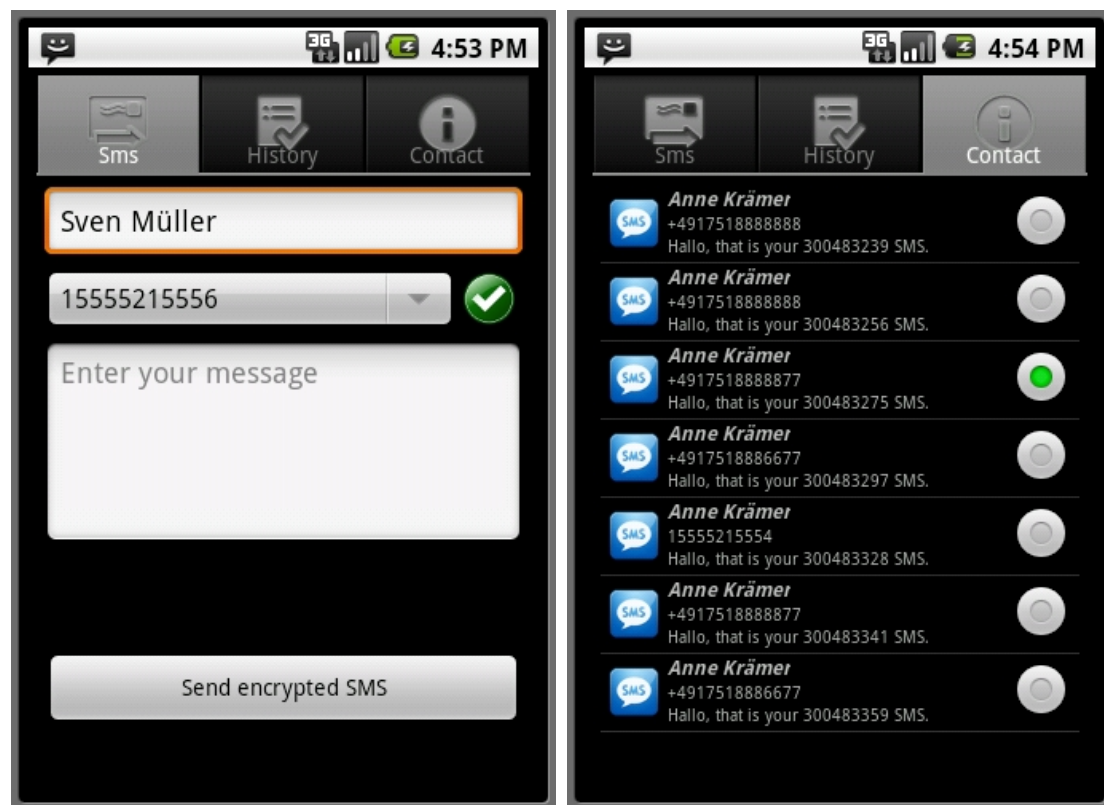


Figure 5.3: Layout: TabView

The only weak spot of this layout is the storing of the data in the **History** tab because if the "Back" button is pushed, the application is terminated, but normally the previous view of the tab should be loaded. In order to solve this problem, the `onBackPressed()` method has to be overwritten. I used a list of "Bundle" containing the information and the ID of the activity to store the history, so that the data can be restored. This took the same time, because the new history has to work correctly in combination with the steps of the Activity Lifecycle (see section 2.5). This means that the history has to be integrated into the functions which are buffering the input information.

In order to present the recent SMS and the contact overview in a user-friendly way, I make use of the **ListView** (see section 4.6.2) integrated in the **TabView** because it is optimized for huge lists of data and the presentation can be changed in almost every way. Each basic view, including images, can be integrated into the presentation of a single view element. This offers a wide range of design possibilities.

### 5.3.4 Support tools for improving the layout and usability

The Android OS offers a wide range of special views, the TabView and the ListView are very popular in application development but there are also other views which improve the usability substantially.

The AutoCompleteTextView offers a simple way to find a specific information in a wide range of data; for that reason I use the AutoCompleteTextView (see section 4.6.3) in the layout of the **Send SMS** tab for selecting the contact. Figure 5.3 shows a picture of the **Send SMS** tab showing an AutoCompleteTextView with an open preselection on the left.

In order to include additional functionalities, I used the context-menu in the SmartCom prototype. In the **Last SMS** and the **History** tab the functions for sorting the SMS are accessible by the context-menu. This menu was chosen, because it is very important that the workflow is not interrupted as that provides short ways for the users. Figure 5.3 shows a picture of an open context-menu of the SmartCom application.

Many applications for older mobile phones had a serious disadvantage: if an event, such as receiving an SMS, happened, the corresponding application was started and the user was interrupted in his work and had no influence on this behavior. Sometimes, input data of another application was lost. In section 1.4.2 the notification service is described. It offers a great flexibility for working with event based applications and that is the reason why I decided to use this service, too.

In the background of the SmartCom application, the receiver is waiting for an incoming, encrypted sms. When the receiver is activated, a notification is created and shown in the top bar of the smartphone. I use the version with the persistent icon in combination with a short vibration (see section 2.4.2) because it does not disturb other people in the periphery, but nevertheless the user will be alerted to the event and can start the application whenever he wants. I personally think that an event based application can not work without using this service and the people who were testing SmartCom, confirmed this opinion.

The final layout is based on a simple construction in combination with the special views described above; it already meets the basic requirements for a user-friendly design. The TabView ensures very short ways for the users. In combination with the ListView and the context-menu, a transparent design with a clear, user-optimized way for presentation of the **Last SMS** and **History** tab was found. There is only one part left to be discussed, the most important one, the **Send SMS** tab of the application.

The use of the AutoCompleteTextView for contact selection provides a clear design, but the position of the input field is important, too. The view for the name, number and status has to be near the best position for data presentation defined



by the golden ratio described in section 4.2. In addition these views will be grouped in an contiguous design to improve the overview of the user (see section 4.3.4). In order to improve this effect the status will be shown by a simple but established symbol (see section 4.3.3), a white arrow on green background symbolized the existence of an secure connection and a white cross on red background symbolized an insecure connection. Many older applications have large button bars, they destroy the transparency of the layout. To avoid this effect, the key exchange function is connected to the status icon and the two functions **Send SMS**/**Send encrypted SMS** are combined into one button. This button changes its label depending on the status of the contact. I chose this solution as the majority of my test subjects did not want to send unencrypted SMS with the application, if they have the possibility to send encrypted ones. In a future version, which may lean towards full organizer functionality, one may have to think about an adjusted solution.

The straightforward layout of the **Send SMS** tab, also fulfills the design principles described in section 4.3. Figure 5.3 shows the final version of the **Send SMS** tab on the left and the final version of the SMS summary of one selected contact on the right.

## 5.4 Final processes

After the description of the technical changes and the design decisions, the final version of the core processes will be presented in this chapter. The reader will get a detailed description of:

1. The login procedure
2. Storing and loading data
3. The key exchange
4. Sending and receiving SMS

### 5.4.1 The login procedure

After loading the contact data, the login screen appears. At the first login into the SmartCom application a password has to be entered and confirmed by entering

it a second time. After this, a global key is created depending on the password. During later logins, the user only has to enter this password to start the application. Figure 5.4 shows a detailed description of the login process:

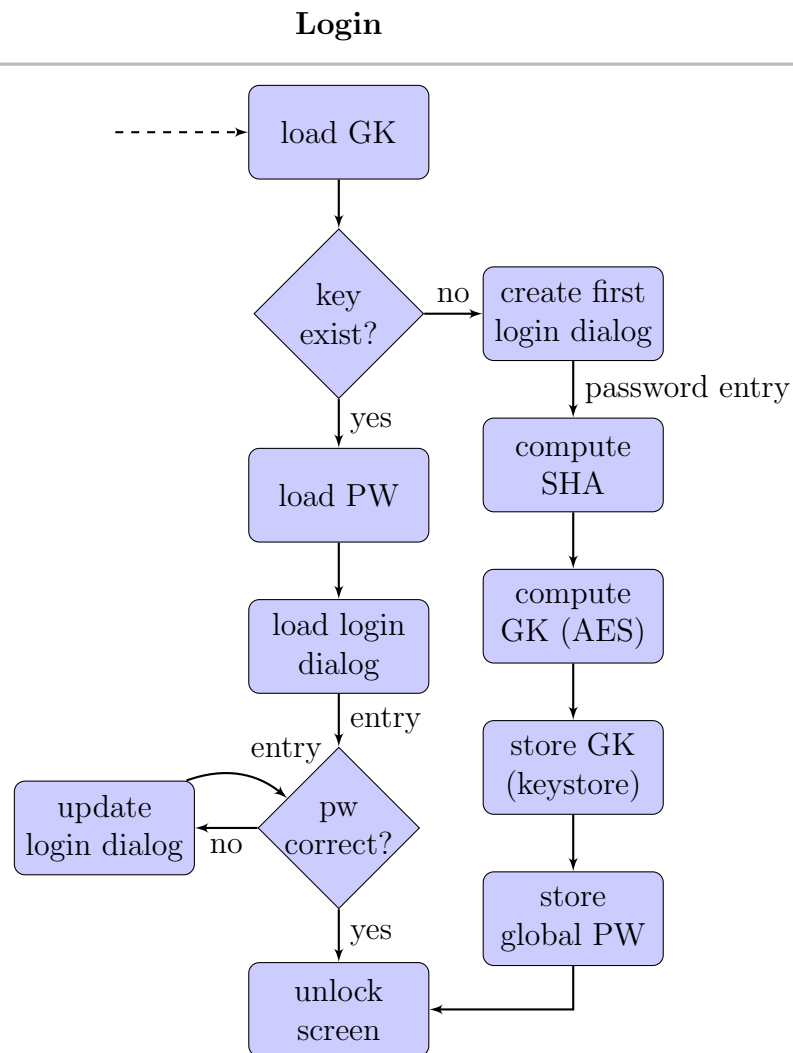


Figure 5.4: *Process: Login procedure*

### 5.4.2 Storing and loading data

The process of the activity is protected against access from the outside by the Android Sandbox Principle (see section 2.3). Moreover, it is also necessary to encrypt all stored data to ensure the best level of security. During the usage of the SmartCom application every SMS and public key will be encrypted with the global key created at the first start. Figure 5.5 illustrates the steps for storing and loading data:

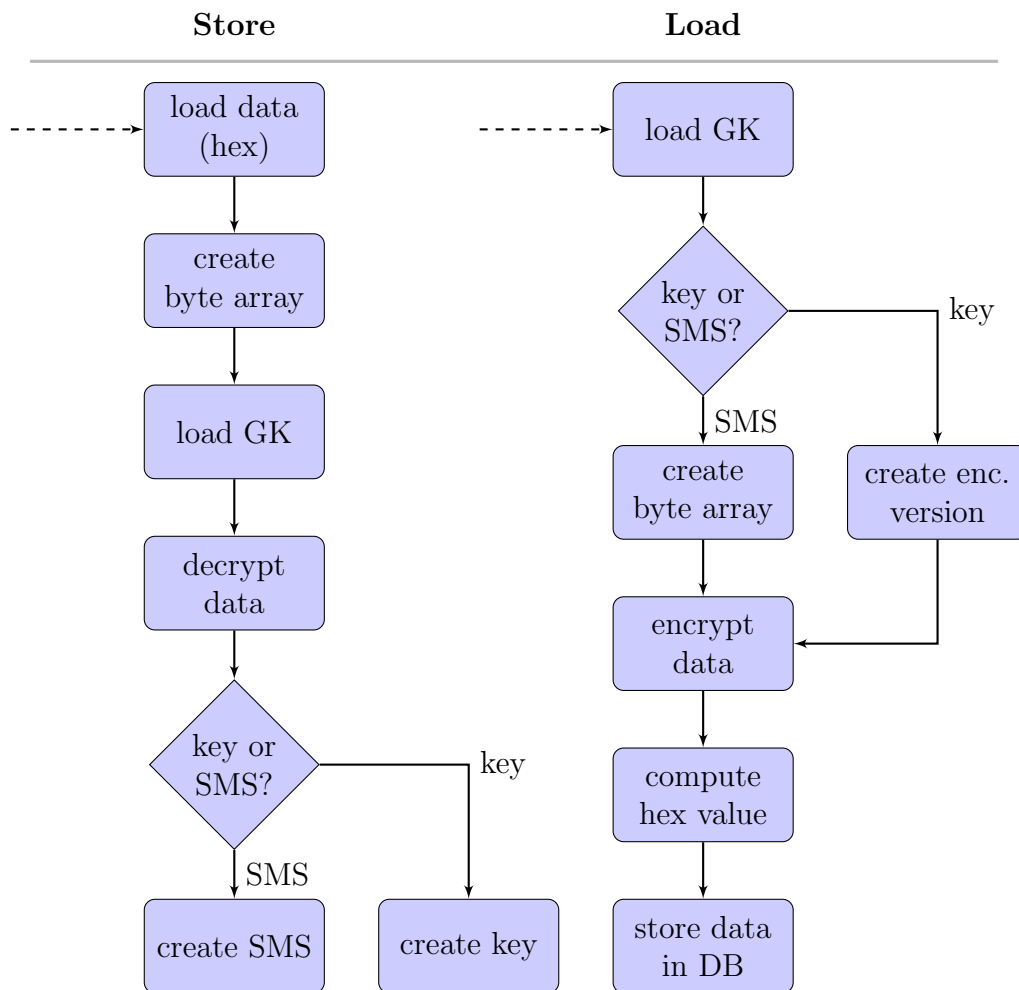


Figure 5.5: Process: Storing and loading data

### 5.4.3 The key exchange

The key exchange is the most important process and is responsible for the key creation to encrypt later SMS. It will also upload the public key of the user on the first connection. Then, the public key of the contact and the trusted certificates for the ssl-connection are loaded and a connection is initialized. On the server side, optionally, the new public key of the user is stored.

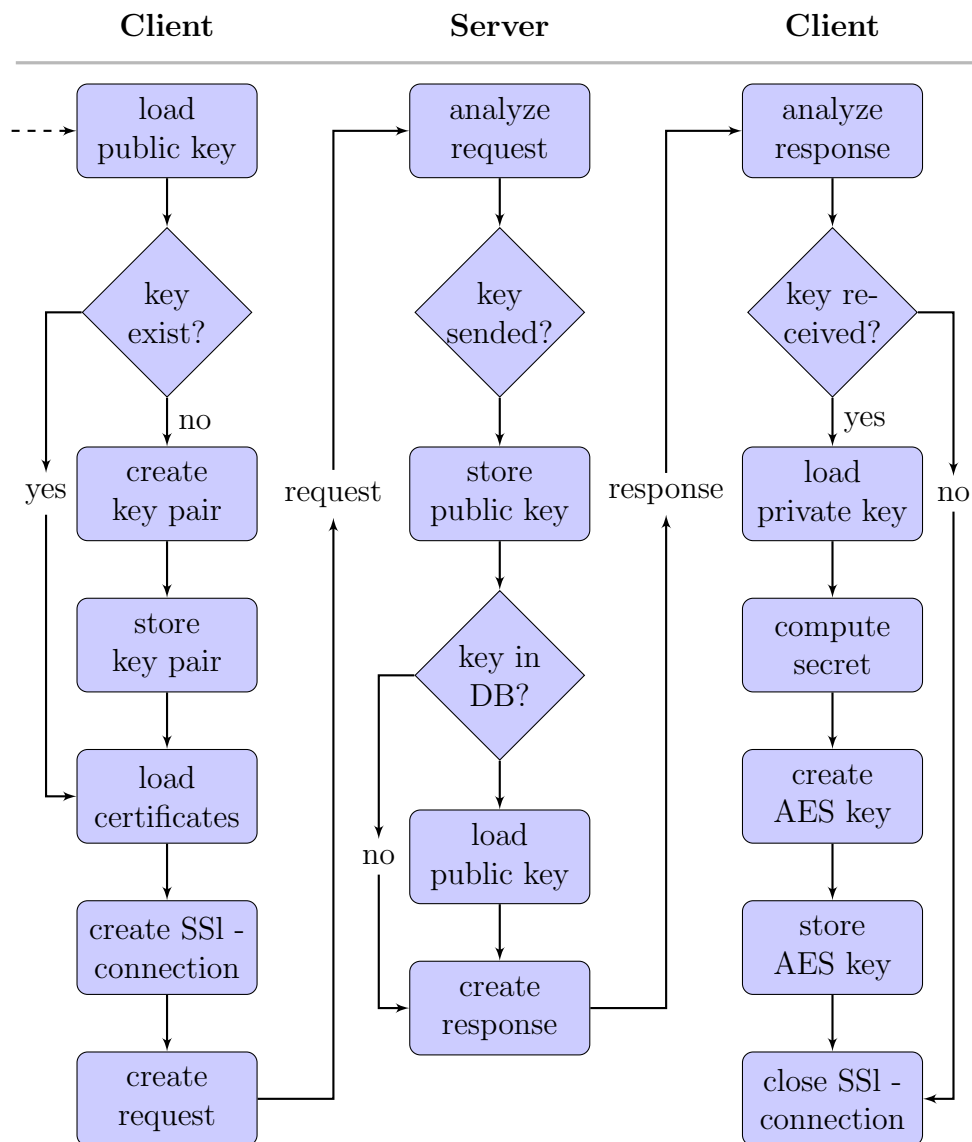


Figure 5.6: Process: Checking for key

The requested public key is then loaded from the database. After that, the response is analyzed on the client side and optionally a new key for encryption is generated. At the end the SSL connection is closed. Figure 5.6 shows a detailed description of the whole process.

#### 5.4.4 Sending and receiving SMS

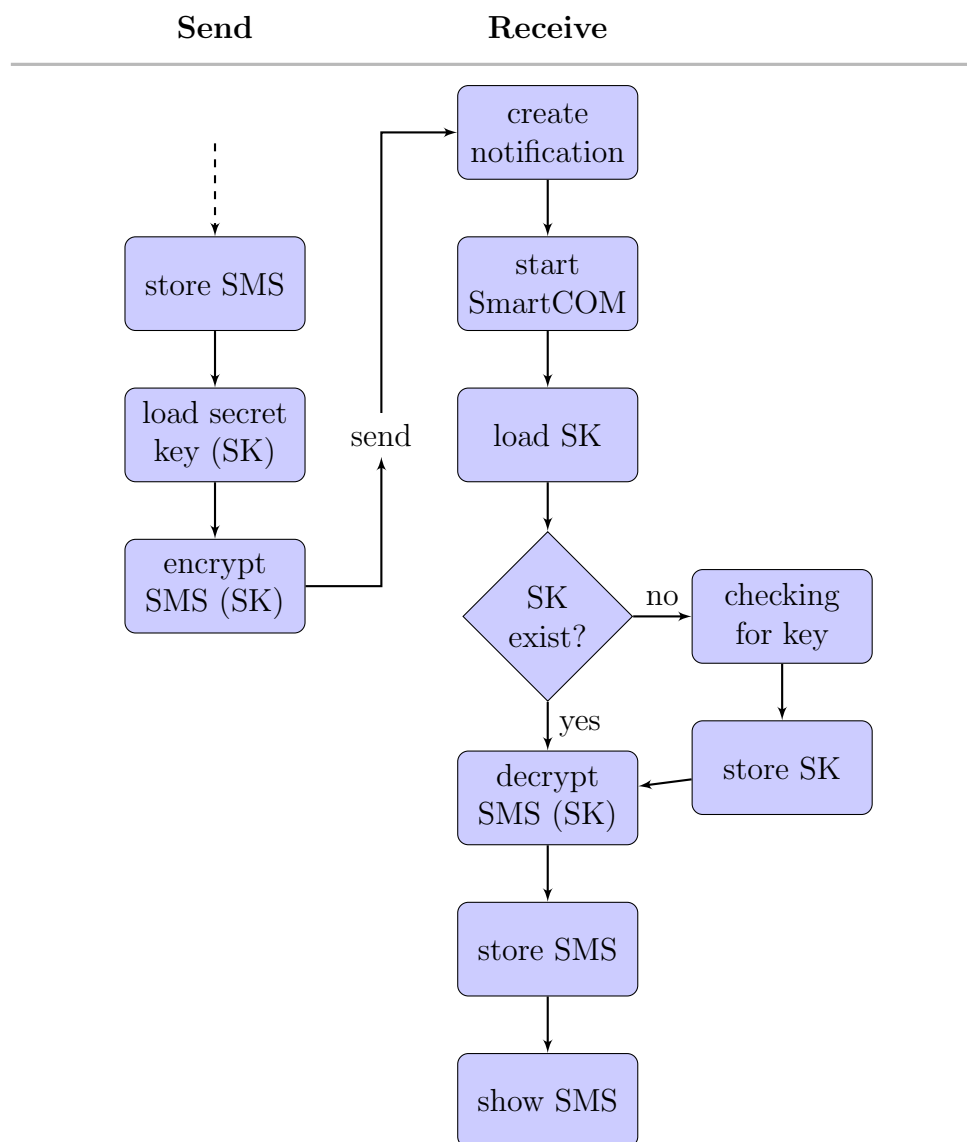


Figure 5.7: Process: Sending and receiving SMS

During the process of sending and receiving SMS, the message is encrypted with the global key and stored in the database of the sending device. Then the secret key of the contact is loaded, the SMS is encrypted and sent to the second device. There the secret key of the sender is loaded and the SMS is decrypted. After that, the SMS is encrypted with the global key of this device and stored in the database. Figure ?? described the process in an detailed way.

## 5.5 Challenges during the development

During the development process, there was only one larger problem to handle, but many small ones. The larger problem was posed by the emulators. At first, it was hard to find out how to handle two of the emulators for testing, because there are some extraordinary errors, which can occur:

- The emulator crashes:
  - Starting multiple emulators simultaneously.
  - Overlapping emulator windows while starting and sometimes also later.
- The emulator gets weird:
  - Too fast usage of Android standard menu.
  - Sometimes by starting the device with older data (day before).
  - Sometimes by using a service or a function starting another processes in the background, which calls not supported functionality.
  - Long running-time can generate a wide scope of various errors.

After some time I learned to handle these errors, but there were still many "ghost-errors" I was chasing, which only occurred because I used the emulator too long or was too fast for the standard menu, etc. Sometimes it was hard to figure out what the problem was, especially when these errors were combined with a programming error. Restarting the emulator every time an error occurs is not the best solution, because it takes several minutes to start the emulators for the tests one by one. Please note that I only described the origins and causes of the errors, for a detailed

description of their remediation would surpass the scope of this thesis as they were too numerous to explain in depth.

In addition to the problems with emulators, there were also some smaller ones, based on insufficient or wrong information:

- Getting detailed information about the supported functionality of the emulators.
- Find a way to run the Bouncy Castle crypto package described at the beginning of the chapter.
- Problems after changing the version of the Android OS although the mechanism I used should be downward compatible.
- The package integrated in the Android OS does not offer the same functionality as the normal java package.

Even at the start of my diploma thesis, the official documentation was very limited, as soon as more detailed information was needed. During the last one and half year, however, the Android OS became very popular and so the documentation becomes more detailed.

In order to work with an SSL connection or, more generally, a client-server architecture it is important to think about the tests. It took me some time to figure out that the emulators also have their own "localhost"; to fix this problem, the IP address of the server has to be used instead of localhost. The problem is also fixed if the server has a dedicated domain name like "www.corsec.de".

The creation of the interface of the application is an another time consuming aspect, because the Eclipse version for Android development has an integrated interface designer, but if a complex view has to be created, it is very hard to get the desired results. Aside from that, it took a lot of time to switch between the two perspectives in Eclipse. After some research, it was much easier to do that by hand. In addition, the source code was much more clearly and readily adaptable to new design ideas.

# Chapter 6

## SmartCOM - A demonstration

After the description of the development and the core process, this chapter will present the final version of the SmartCom prototype. The reader will get an illustrated and detailed overview of the workflows.

### 6.1 Login

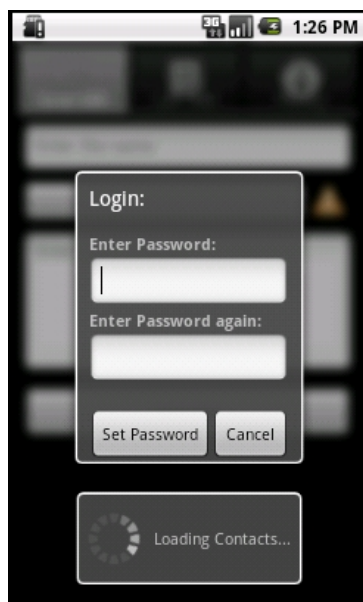


Figure 6.1: *First login screen*

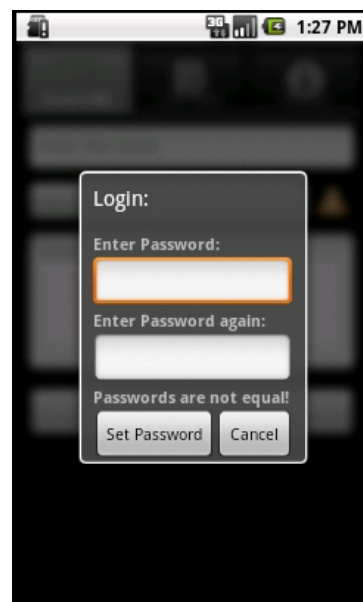


Figure 6.2: *First login error*



In order to work with the SmartCom application, the user has to login. First of all, the contact data is read-out from the phonebook. During the loading time a dialog is displayed to inform the user. Then the user has to select and enter his password. Then the equality of the password is checked and if it matches, the global encryption key is derived (see 5.4.1) and stored in the database. Figure 6.1 shows the loading dialog and figure 6.2 the login error on the right which can occur.

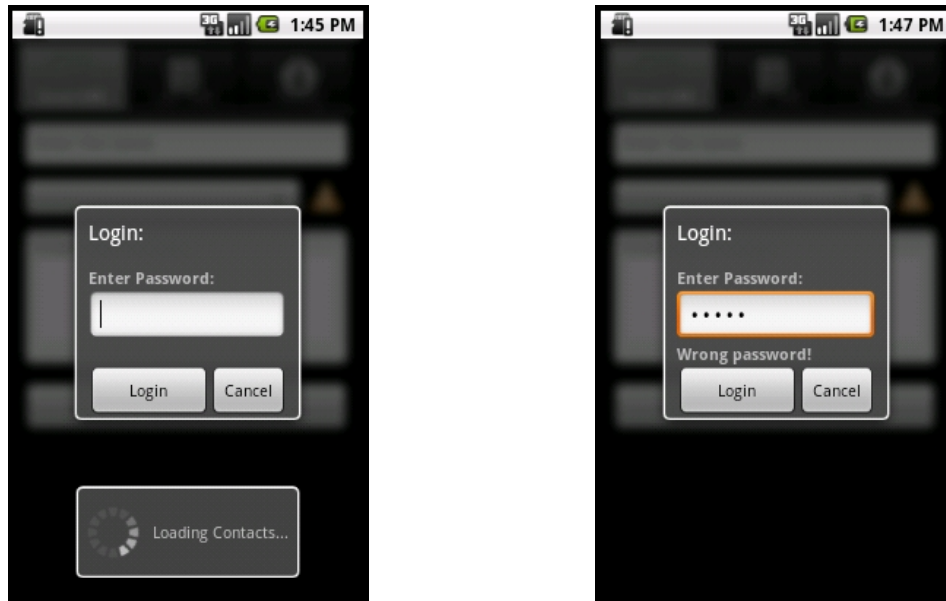


Figure 6.3: Standard login screen

During the standard login the contact data is also read-out. Then the user has to enter his password and if it is valid, the SmartCom application starts and the Send SMS tab will be in front. Figure 6.3 shows the standard login.

## 6.2 Key Exchange

After the login, the TabView layout (see 4.6.2) of the SmartCom application is displayed and the Send SMS tab is in front. As the contact data is read-out, the users can choose one of their contacts in the AutoCompleteTextView (see 4.6.3) at the top. Then the users, let me call them Anne and Tim, can start the key exchange (see 5.4.3) by clicking on status symbol on the right of the name field.

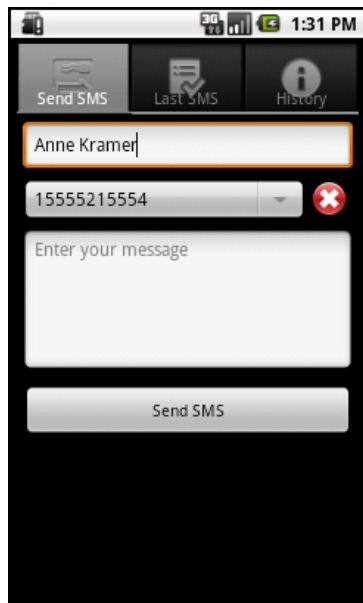


Figure 6.4: Key exchange part 1

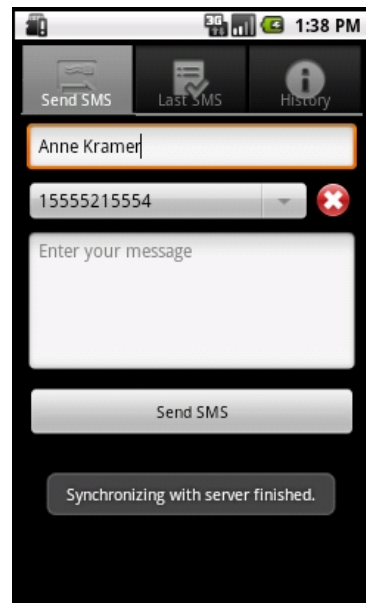


Figure 6.5: Key exchange part 2

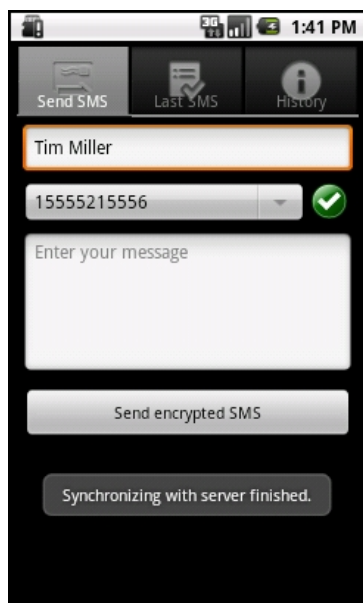


Figure 6.6: Key exchange part 3

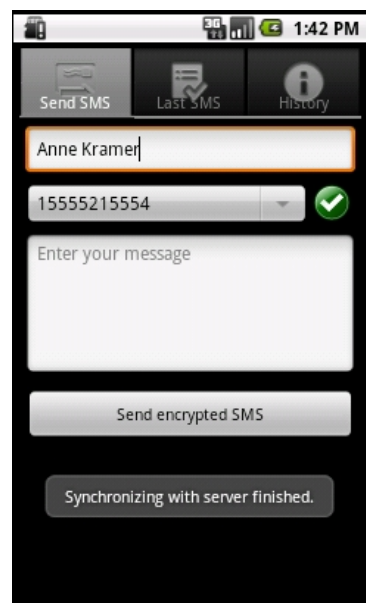


Figure 6.7: Key exchange part 4

Tim selected Anne (Figure 6.4) and starts the exchange. Since, Anne has not started any exchange before, no public key is available on the server and Tim

gets the notification, that the synchronization is finished, but the status will leave unchanged (Figure 6.5). Then Anne wants to send an encrypted SMS and in order to do that, she also starts the exchange. In contrast to Tim, she received a public key, because Tim has started an exchange before and his public key was uploaded. The status switches to a checkmark on green background and the button label changes to "Send encrypted SMS" (Figure 6.6). Later, Tim starts the exchange again and can receive the public key of Anne (Figure 6.7). Even if Tim does not start the key exchange again, the public key will be downloaded automatically by receiving an encrypted SMS from Anne. Henceforth, Anne and Tim can transmit their informations via encrypted sms.

## 6.3 Send SMS

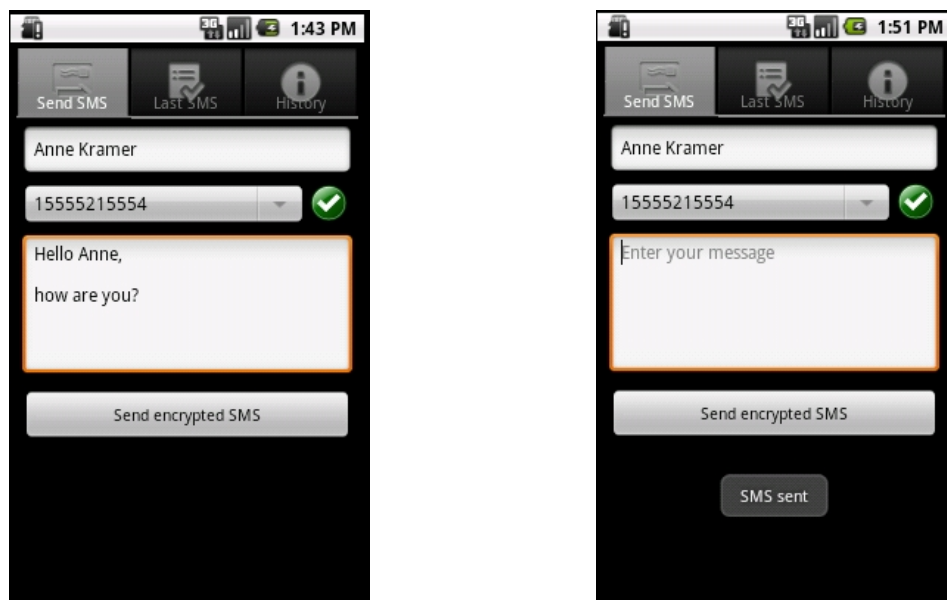


Figure 6.8: *Send encrypted SMS*

After establishing a common secret, encrypted sms can be sent. This takes not more time as for a standard SMS. The user has to enter the text in the text field and can send it by clicking on the button "Send encrypted SMS". After sending an SMS, the user is informed about the status by a notification on the screen and the text field is cleared. Figure 6.8 shows the simple workflow.

## 6.4 Receive SMS

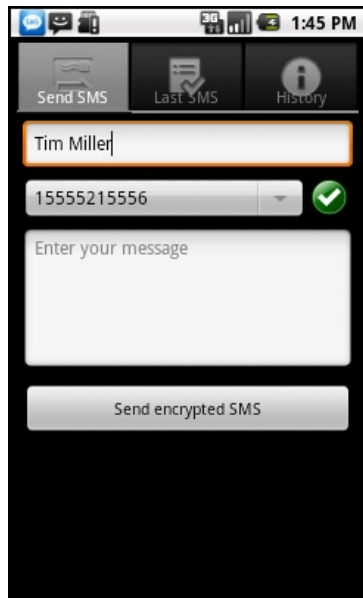


Figure 6.9: *Receive SMS part 1*

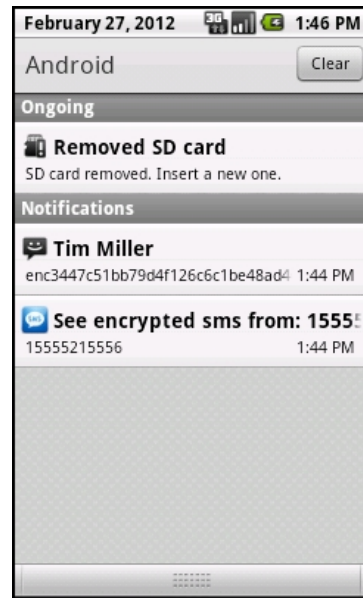


Figure 6.10: *Receive SMS part 2*

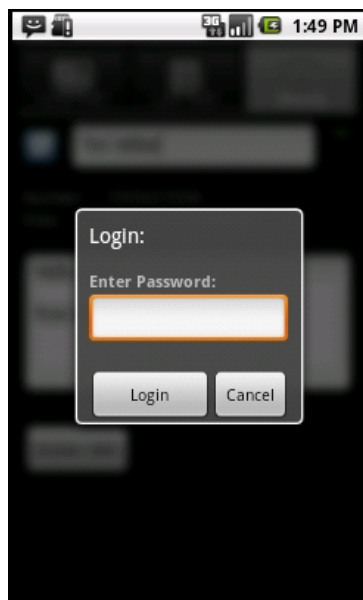


Figure 6.11: *Receive SMS part 3*

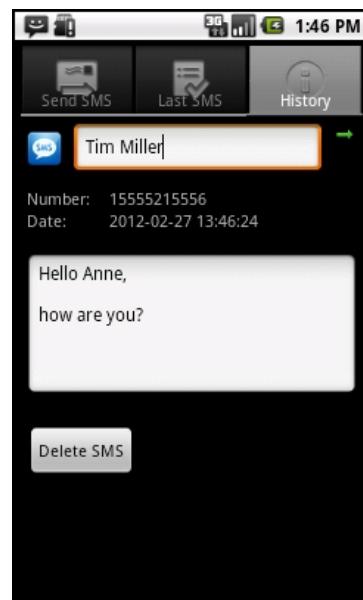


Figure 6.12: *Receive SMS part 4*

At the time when the user receives a SMS, a notification (see 2.4.2) is shown on top in the notification bar (Figure 6.9). Nothing changes to the user, he can continue his work uninfluenced. In order to open the encrypted SMS, the user can start the SmartCom application by opening the notification menu and clicking on the corresponding notification (Figure 6.10). Then the application starts or comes to the front depending on if the user was working with the application or not. When the application has to start, the user has to enter his password in the login screen (Figure 6.11), otherwise if the user was working with SmartCom, the History tab will open and the received SMS is decrypted and displayed (Figure 6.12).

## 6.5 Last SMS

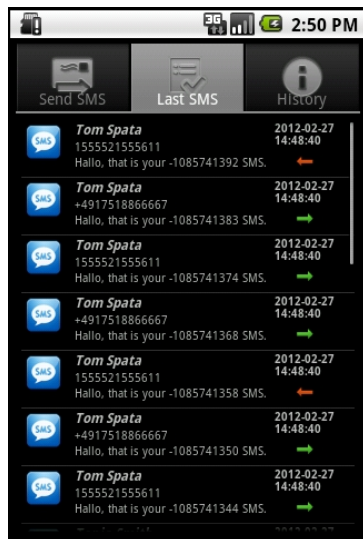


Figure 6.13: Last SMS part 1

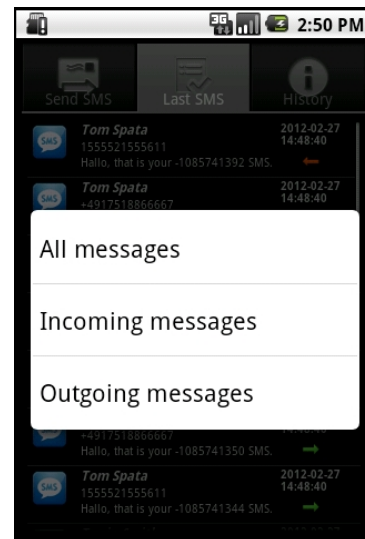


Figure 6.14: Last SMS part 2

The Last SMS tab offers the user the possibility to see on one click which SMS he has received in the last time (Figure 6.13) ordered by date. The SMS can be easily distinguished between incoming and outgoing SMS by the red or green arrow at the right of the ListView element (see 4.6.2). Aside from this, the user can also change the view by opening the underlying menu via a long click on a list element to display only the incoming or outgoing SMS (Figure 6.14).

## 6.6 History

The History tab contains all messages of the contacts. At first the contact overview is in front (Figure 6.16).

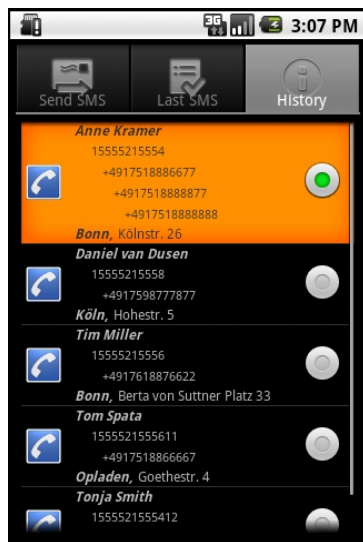


Figure 6.15: History part 1

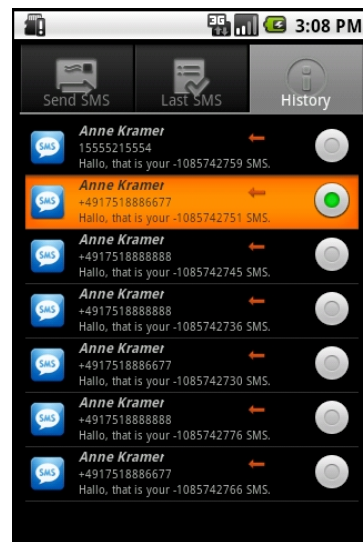


Figure 6.16: History part 2

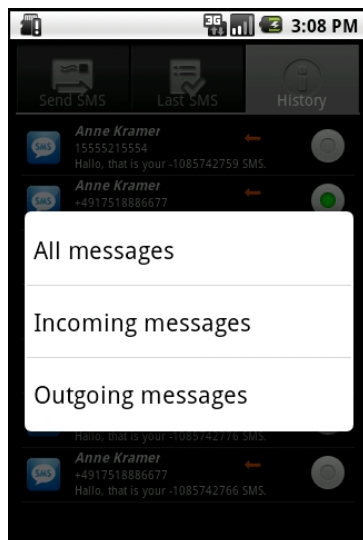


Figure 6.17: Demo: History part 3

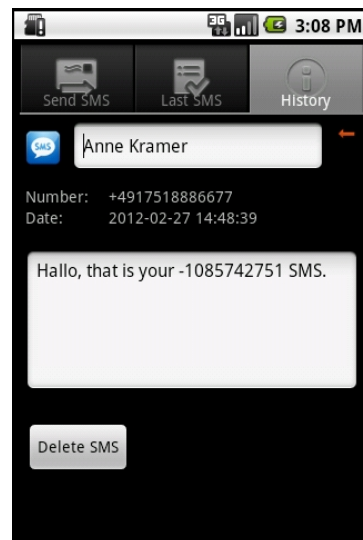


Figure 6.18: Demo: History part 4

In this view, the user can choose a contact by clicking on it. Then the view will change and the SMS of the selected contact are displayed on the screen ordered by date (Figure ??). Via a long click on a message, the user can open the underlying menu and distinguish between incoming and outgoing SMS (Figure 6.17). When the user has found the desired SMS, he can open it by a click. The next view displays the whole SMS and the user has the possibility to delete it (Figure 6.18). The History tab is a very basic version of an SMS organizer. In future versions more functionality has to be added.

# Chapter 7

## Conclusion / Further Work

The SmartCom prototype presented in chapter 6 shows that a high level of security and good usability can be combined in one application. The application is compatible to all Android versions higher than 2.1 which means that nearly 50% of all smartphone user should be able to use this application and communicate securely via text messages.

The **History** tab of the application is a small, basic version of an SMS organizer. In future versions of the application, this functionality will offer a lot of potential for extension to increase the attractiveness of application. Offering more possibilities for sorting, searching specific SMS and deleting multiple SMS is one option to do that. Another option is to include the key-exchange functionality into the ListView of the contact overview in combination with the contact symbol.

As the read-out mobile number of the smartphone is currently used as a password for the authentication of the client against the server, the password can not be changed. This is essentially an adaptation of the password authentication used e.g. in online banking. In the future, this authentication could be changed to another password based mechanism or a process based on certificates which would allow the user to update their authentication credentials. A prerequisite for the latter solution would be, that every user has a certificate from a trusted key authority (e.g. as part of an identity card).

The application was developed in an emulator based development environment. If the application can be tested in a live environment, it would be easy to extend the application for MMS encryption, because the emulator does not support the MMS functionality.

Voice call encryption is another functionality which would be a powerful extension of the application; unfortunately the Android API offers no deep level access to modify the voice channel before it is sent. At that time, the only way to encrypt a voice channel is via Voice over IP by using the internet connection. However,



this solution needs a dedicated server to establish a connection between two users, because the smartphones have no fixed IP addresses and the server handles this problem. The described application would be very similar to Skype extended with an encryption mechanisms. Presently, there is no other possibility to encrypt the voice channel, but during the next years the IP address will be reformed. Maybe with more available IP addresses, every smartphone has its own fixed adress and Voice over IP can be implemented directly.

# Bibliography

- [Anda] *Android Activity Lifecycle*. Webpage. <http://developer.android.com/reference/android/app/Activity.html>
- [Andb] *Android Contact Provider*. Webpage. <http://developer.android.com/reference/android/provider/ContactsContract.html>
- [Andc] *Android Notifications Service*. Webpage. <http://developer.android.com/guide/topics/ui/notifiers/notifications.html>
- [Andd] *Android SSL*. Webpage. <http://developer.android.com/reference/javax/net/ssl/package-summary.html>
- [Ande] *Android Telephony Service*. Webpage. <http://developer.android.com/reference/android/telephony/package-summary.html>
- [Andf] *The AndroidManifest.xml File*. Webpage. <http://developer.android.com/guide/topics/manifest/manifest-intro.html>
- [ANS] *ANSI X9.62:2005*. Webpage. <http://webstore.ansi.org/RecordDetail.aspx?sku=ANSI+X9.62:2005>
- [Apa] *Apache Tomcat*. Webpage. <http://tomcat.apache.org/>
- [Ber10] BERNDT, Thomas M.: *CryptCOM - Insuring secure communication on arbitrary GSM phones by applying strong cryptography*, Universität Bonn, Diplomarbeit, 2010
- [BJS07] BARKER, Elaine ; JOHNSON, Don ; SMID, Miles: *NIST Special Publication 800-56A*. Webpage. [http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A\\_Revision1\\_Mar08-2007.pdf](http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf). Version: March 2007. – NIST Special Publication 800-56A

- [Bon] BONN, Institute for Computer Science III University o.: *Scotland Yard - to go!* Webpage. <http://sam.iai.uni-bonn.de/projects/amoga/>
- [Bor08] BORT, Dave: *Android is now available as open source.* Webpage. <http://web.archive.org/web/20090228170042/http://source.android.com/posts/opensource>. Version: October 2008
- [Bou] *The Legion of the Bouncy Castle.* Webpage. <http://www.bouncycastle.org/>
- [BP10] BECKER, Arno ; PANT, Marcus: *Android 2 - Grundlagen und Programmierung (2te - Auflage).* Heidelberg : dpunkt, 2010. – ISBN 978-3898646772
- [Bul00] BULMAN, Philip: *Commerce Department Announces Winner of Global Information Security Competition.* Webpage. [http://www.nist.gov/public\\_affairs/releases/g00-176.cfm](http://www.nist.gov/public_affairs/releases/g00-176.cfm). Version: October 2000
- [Bun09] *Elliptic Curve Cryptography.* Webpage. [https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03111/BSI-TR-03111\\_pdf.pdf?\\_\\_blob=publicationFile](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR03111/BSI-TR-03111_pdf.pdf?__blob=publicationFile). Version: April 2009. – Technical Guideline TR-03111
- [BWBG<sup>+</sup>06] BLAKE-WILSON, S. ; BOLYARD, N. ; GUPTA, V. ; HAWK, C. ; MOELLER, B.: *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS).* Webpage. <http://www.ietf.org/rfc/rfc4492.txt>. Version: May 2006. – RFC 4492
- [DA99] DIERKS, T. ; ALLEN, C.: *The TLS Protocol, Version 1.0.* Webpage. <ftp://ftp.ietf.org/rfc/rfc2246.txt>. Version: January 1999. – RFC 2246
- [DH06] DIFFIE, Whitfield ; HELLMAN, Martin E.: *New Directions in Cryptography.* Webpage. <http://www.cs.jhu.edu/~rubin/courses/sp03/papers/diffie.hellman.pdf>. Version: May 2006
- [DR02] DAEMEN, Joan ; RIJMEN, Vincent: *The Design of Rijndael. AES -*

- The Advanced Encryption Standard*. Berlin, Heidelberg, New York : Springer, 2002. – ISBN 3–540–42580–2
- [DR08] DIERKS, T. ; RESCORLA, E.: *The Transport Layer Security (TLS) Protocol Version 1.2*. Webpage. <http://www.ietf.org/rfc/rfc5246.txt>. Version: August 2008. – RFC 5246
- [Ecla] *Android Development with Eclipse*. Webpage. <http://www.eclipse.org/resources/resource.php?id=516>
- [Eclb] *Eclipse IDE for Java EE Developers*. Webpage. <http://www.eclipse.org/downloads/moreinfo/jee.php>
- [FKK96] FREIER, Alan O. ; KARLTON, Philip ; KOCHER, Paul C.: *The SSL Protocol, Version 3.0*. Webpage. <http://web.archive.org/web/20080206214535/http://wp.netscape.com/eng/ssl3/draft302.txt>. Version: November 1996. – Transport Layer Security Working Group (IETF)
- [Gar10] *Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent*. Webpage. <http://www.gartner.com/it/page.jsp?id=1848514>. Version: November 2010
- [Gar11] *Gartner Says Sales of Mobile Devices Grew 5.6 Percent in Third Quarter of 2011; Smartphone Sales Increased 42 Percent*. Webpage. <http://www.gartner.com/it/page.jsp?id=1848514>. Version: November 2011
- [Hof10] HOFFMANN, Manuela: *Modernes Webdesign - Grundgestaltungsprinzipien, Webstandards, Praxis*. 53227 Bonn, Rheinwerkallee 4 : Galileo Press, 2010. – ISBN 978–3–8362–1502–2
- [LV00] LENSTRA, Arjen K. ; VERHEUL, Eric R.: Selecting Cryptographic Key Sizes. In: *Public Key Cryptography*, 2000
- [NIS02] NIST: FIPS 180-2: Announcing the Secure Hash Standard / Information Technology Laboratory, National Institute of Standards and Technology. Version: August 2002. <http://csrc.nist.gov/fips180-2>

- nist.gov/publications/fips/fips180-2/fips180-2.pdf. 2002.  
– Forschungsbericht
- [NIS09] NIST: FIPS 186-3: Digital Signature Standard (DSS) / Information Technology Laboratory, National Institute of Standards and Technology. 2009. – Forschungsbericht
- [NSA05] *NSA Suite B Cryptography*. Webpage. [http://www.nsa.gov/ia/programs/suiteb\\_cryptography/index.shtml](http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml). Version: 2005
- [NSA09a] *The Case for Elliptic Curve Cryptography*. Webpage. [http://www.nsa.gov/business/programs/elliptic\\_curve.shtml](http://www.nsa.gov/business/programs/elliptic_curve.shtml). Version: Januar 2009
- [NSA09b] *Suite B Implementer's Guide to NIST SP 800-56A*. Webpage. [http://www.nsa.gov/ia/\\_files/SuiteB\\_Implementer\\_G-113808.pdf](http://www.nsa.gov/ia/_files/SuiteB_Implementer_G-113808.pdf). Version: July 2009
- [pro] *Protecle*. Webpage. <http://portecle.sourceforge.net/>
- [SEC] *SECG Released Standards*. Webpage. [http://www.secg.org/index.php?action=secg,docs\\_secg](http://www.secg.org/index.php?action=secg,docs_secg)
- [SG03] SYME, Matthew ; GOLDIE, Philip: *Optimizing Network Performance with Content Switching: Server, Firewall and Cache Load Balancing*. Prentice Hall, 2003 <http://www.informit.com/store/product.aspx?isbn=0131014684>. – ISBN 978-0131014688
- [Sta98] STALLINGS, William: *The Internet Protocol Journal - Volume 1, No. 1: SSL: Foundation for Web Security*. Webpage. [http://www.cisco.com/web/about/ac123/ac147/archived\\_issues/ipj\\_1-1/ssl.html](http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_1-1/ssl.html). Version: June 1998
- [Tha] *Thawte - Online-Sicherheit, der Millionen in aller Welt vertrauen*. Webpage. <http://www.thawte.de/?sl=t88700282810000007&gclid=CKmZl9a2ka4CFUwf3godbyswhQ>

# Index

## A

ADT plugin ..... 46  
 AES.....23–28, 47  
   AddRoundKey ..... 25–26  
   key generation ..... 25  
   MixColumns.....25, 27–28  
   Rijndael-Algorithm ..... 24  
   S-Box ..... 26, 28  
   ShiftRows ..... 25–28  
   SubBytes ..... 25–26, 28  
 Android  
   activity.....14, 51, 55  
   Activity Lifecycle ..... 12–14, 51  
   architecture.....7–8  
   Contact Provider.....10–11  
   Dalvik Virtual Machine.....8–9  
   integrated systems.....10–14  
   Manifest ..... 9, 10, 12, 44  
   Notification ..... 11, 52  
   Notification Service ..... 11, 52  
   operating system ..... 6–14  
   Protecle Principle.....55  
   Sandbox Principle ..... 9  
   Telephony Package.....12  
 AndroidManifest.....9  
 ANSI.....19  
 ARM-Processors.....8  
 AVD ..... 46  
 AVD manager ..... 46

## B

Bouncy Castle crypto library..47, 59

## D

Dalvik Debug Monitor Server.....47  
 DES.....24  
 Diffie-Hellman ..... 19–20, 47, 48  
 DLP.....18  
 domain parameters ..... 15  
 DSA.....21

## E

Eclipse  
   Android.....46, 59  
   Java EE ..... 45  
 Elliptic curve.....15–19  
   domain parameters ..... 19  
   ECDLP ..... 18, 20  
   NIST curves.....18–19  
   point of infinity ..... 16  
 Elliptic curve cryptography 15–20, 47  
 embedded systems.....7, 8

## F

Finite field ..... 17  
   binary field ..... 19  
   prime field ..... 19  
 finite field ..... 15, 17, 18  
   binary field ..... 18  
   prime field ..... 18

## G

Galois field ..... 27  
 Global key ..... 54, 58  
 Golden ratio.....53  
 golden ratio ..... 35–36

- I**
- iOS.....6
- IP address.....59
- iPhone.....6
- J**
- Java.....4
- JVM Apache Harmony.....8
- K**
- key-agreement.....19
- Koblitz curve.....19
- L**
- Linux.....9
- M**
- Man-in-the-Middle attack.....20, 48
- menu.....39–40
  - context-menu.....39, 52
  - option-menu.....39
- N**
- NIST.....18, 19, 21, 24
- NSA.....19, 21
- O**
- OSI-Layer-Model.....29
- R**
- Receiver.....52
- RSA.....18, 32
- S**
- SECG.....19
- server-client architecture...44, 46, 59
- servlet.....50
- SHA-256.....21–23
  - compression function.....22–23
  - message schedule.....21–22
- SmartCom.....43, 44, 46–58, 65, 68
  - key exchange.....48–49
  - key server.....50
  - processes.....53
    - key exchange.....56–57
    - loading data.....55
    - login procedure.....53–54
    - receiving SMS.....58
    - receiving sms.....57
    - sending SMS.....58
    - sending sms.....57
    - storing data.....55
- SSL.....28–33
  - Alert Protocol.....32
  - Application Data Protocol....32
  - architecture.....29–30
  - Change Cipher Spec Prot..31–32
  - Handshake Protocol.....30–32
  - Record Protocol.....32–33
- SSL connection....48, 50, 56, 57, 59
- Suite-B.....19
- Symbian.....6
- T**
- TCP/IP-Network.....29
- Thawte.....49
- Tomcat server.....45, 46
- V**
- view
  - AutoCompleteTextView..42, 52, 61
  - ListView....40–41, 51, 52, 65, 68
  - TabView.....39–41, 50, 52
  - Voice over IP.....68, 69
- W**
- Weierstrass Equation.....16