CS344 - Final Report

Smart Phone Cryptography

A comparison of techniques for encrypted data communication

Tom Nicholls - 1006007

Discrete Mathematics - Year 3

Marcin Jurdzinski

Copy 2

April 24, 2013

Abstract

Protecting sensitive or secret information has always been an important issue, particularly with the increased popularity and usability of smartphones. The aim of this project is to make an original conclusion about the current state of secure smartphone data communication applications, through an implementation and analysis of currently used, researched, techniques. It was found that the currently used cryptographic techniques are appropriate for the current state of smartphone usage and capabilities. These techniques, however, should be combined to create a new application, with possible improvements available in the event that the requirements of the current cryptographic schemes surpass the available resources of the average smartphone.

Keywords: Cryptography, Encryption, Decryption, Smartphone, Android, Application, Network

Contents

Chapter 1

Introduction

1.1 Background

To start this project report, all relevant background information will be presented to allow the reader to fully understand all aspects of the project. An overview of basic techniques and topics will be given, with a greater explanation for more specialised areas.

1.1.1 Smartphones

In this section, the basics of the smartphone will be introduced and described. As smartphones are a prominent part of modern culture, and therefore known in some degree to everybody, only an overview will be given. The book by Sarah Allen et al. [1] was used as reference material.

'Mobile' or 'Cell' phones have been available for commercial use since the beginning of the 1980s. The most popular functions of these phones are for telephone calls, text or multi media message sending or even basic internet access and games. Higher-end phones, universally named the 'smartphone' and available from the mid-90's, provide the same basic features (telephone calls, messaging etc) but with a plethora of added functions such as full internet access, as well as increased computing capabilities through more powerful processors and other hardware. Smartphones also utilise the QWERTY keyboard and a larger, higher-resolution, screen.

Compared to desktop computers, smartphones have a diverse set of operating systems, determined by the manufacturer of the smartphone. Smartphone operating systems include; Apple iPhone iOS [2], Windows Phone [3] and Google Android [4]. Each operating system, unlike that of a desktop, determines which programming language a developer must use if they are to develop an app for a particular smartphone. This leads to a separate application marketplace, a database where users can download and install new

applications, for each smartphone operating system. Whilst applications can be developed in a cross-platform manner, through the use of HTML and CSS, this is not yet standard, so the various marketplaces are individual in the applications that are on offer. The Android operating system, developed by Google and mostly found on devices by Samsung, HTC or Google themselves, use the Google Play marketplace [5] for application distribution with applications being developed using the Java programming language.

The message sending capabilities of smartphones is the feature that is most important for this project, whether it be using the in-built short messaging service (SMS) or through the use of an installed application. On average, the message is of a short length and can be described as a simpler form of email exchange.

1.1.2 Cryptography

Taken from the book by Richard A. Mollin [6], the meaning of cryptography is 'the study of methods for sending messages in secret (namely, in enciphered or disguised form) so that only the intended recipient can remove the disguise and read the message (or decipher it).' Many keywords are used in the study of cryptography and help to give an introduction to the field;

- Plaintext - The original message, input by the initiating user.

- Ciphertext - The disguised message, created using the plaintext.

- Encryption - The process of transforming the plaintext into the ciphertext.

- Decryption - The process of turning the ciphertext into the original plaintext. Accomplished by the recipient, who has the knowledge to remove the disguise.

- Cipher - The Method for enciphering and deciphering.

- Cryptanalysis - The study of mathematical techniques for attempting to break the cryptographic methods.

- Cryptographic Key - A tool for encryption or decryption

This shows that basic cryptography has the following form:

$$\textbf{Plaintext} \rightarrow \textit{Encryption} \rightsquigarrow \textbf{Ciphertext} \rightsquigarrow \textit{Decryption} \rightarrow \textbf{Plaintext}$$

Two very basic examples of cryptography, which can and have been expanded into many different, more elaborate techniques, are the substitution and transposition ciphers. Substitution ciphers replace symbols in the plaintext with other symbols, using a given rule

(the key), to produce the ciphertext. For example, the key may be; a → q, b → f, c →
m, and so on. A transposition cipher, on the other hand, transposes the places in which
the plaintext letters are situated. This means that no new letters are introduced. The
key for this cipher is a permutation that describes how the letters should be transposed.
For a plaintext that consists of 10 letters, the key could be:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 5 & 7 & 6 & 4 & 9 & 10 & 3 & 8 \end{pmatrix}$$

where the symbol in the position number in the top row, is replaced by the symbol
in the position number below it (in the second row). Using these techniques as a basis,
other more advanced cryptographic schemes can be created and are defined by the cipher
(and key) that is used for the particular method.

Cryptography has been used in some form or another to exchange secret message
throughout history. For example, the first documented use of cryptography in a military
setting was by the Spartans in 475 B.C. In the modern age, cryptography is most com-
monly used when sending or storing data using a computer system, such as sending an
email across a network or saving a file on a company server.

1.2   Motivation

Protecting important and secret information is a big issue, particularly when data such
as bank accounts details or private work related conversations are taking place or being
exchanged. When new devices or computer systems are developed and deployed, the
security of that system is always considered and studied. Smartphones, on the other hand,
have a substantial amount of user-input in the form of publicly developed applications.
With smartphones becoming the most popular tool for communicating and completing
other tasks with the transfer of sensitive information, combined with the fact that there
are a vast number of tools for completing such tasks, the study of the strength and
current state of security of smartphones is an ever evolving field. This creates a number
of questions, the answers to which form the basis of this project:

- Do applications that facilitate the secure transfer of messages exist?

- What are the cryptographic techniques behind these applications?

- Are there other techniques which could also be used for this purpose?

- What is the performance of these styles applications in comparison to each other?

- Can these applications or methods be improved upon?

- Can a conclusion be made about the current state of secure message communication and its future?

The results of this project will contribute an original conclusion to the field of cryptography within smartphones. Preliminary research showed that the questions set out to be answered in this project have not been answered before for the current, present-day state of smartphones. This is important because even in the last few years significant improvements and developments have been made concerning the design and processing power of smartphones, which has had a tremendous effect on smartphone usability, for example the more advanced and widely available use of GPS in direction planning.

1.3 Scope and Limitations

It is important to set out in the introduction to this project report what the scope of the project is and to describe any limitations that have been placed on the project.

Firstly, as will be detailed later in this report, the choice was made that any developed software will be in the Java programming language and that the Android operating system will be the focus of study and application development. This is so that the project can be accomplished within the given time frame, to the standard required and with hardware that is more easily available. It also ensures that the project does not try to encompass an area that is so huge that any conclusion loses focus and therefore its value.

This project also focuses on the use of cryptography in sending secure messages between users as opposed to other issues regarding smartphone security. A few examples of other forms of security related to smartphones could be; a study of stored data protection, techniques to retrieve data from a smartphone wirelessly or protecting a smartphone from attacks or viruses. Narrowing the scope of the project in this way means that a full solution and conclusion can be found for this particular aspect, as opposed to giving brief or vague conclusions for a number of aspects. Cryptography, particularly in this setting, also allows for a project that encompasses both computer science and mathematics and therefore is relevant to both aspects of the Discrete Mathematics course.

Lastly, the goal of this project is not to design and develop a product that meets industry and user requires and can be sold or released to the general public. A project of that type would focus more on human-computer interactions and product design and marketing as opposed to focusing on the mathematically based computer science areas of smartphones and cryptography. Therefore, the project can be categorised as a combination of both a research and a software development based project, with each complimenting the other.

1.4 Issues

Every project is faced with issues regarding the projects final goals and outcomes, or the techniques or methods used to reach these outcomes. This section will discuss the issues faced or considered with this project.

1.4.1 Legal

Legal issues are the main issues that need to be considered within this project. As a large section of this project is research based, it needs to be ensured that all materials used are correctly and appropriately referenced. Furthermore, as this project is centred on the issues of computer security, in the form of cryptography, legal issues need to be discussed [7] [8]. As long as any material that is encrypted is legal in its own rights and that only data that is solely owned by myself is used in the cryptographic processes, then any legal issues are negated. Also, the resulting product facilitates the secure communication of messages which, in the wrong hands, could be used to aid a number of illegal operations. To avoid the direct use of the final application for illegal purposes, it will not be published to the Google Play application marketplace and will therefore not be available to the public. This also means that a disclaimer or end-user license agreement (EULA) will not be necessary, which would require the consultation of a professional lawyer. If the software created in this project was obtained by a malicious user then any data required by the police could and would be given to break the illegal encryption [9].

1.4.2 Ethical

In almost all modern technical advances, ethical issues can be found, posing unique problems depending on the perception or views of the topic by various groups or persons. For this topic, the legal issues presented above can also be viewed as ethical issues. Should an application be developed, or a field be advanced, that could be used to facilitate illegal operations? This question, and many like it, are discussed constantly by a variety of people from a variety of backgrounds, both academic and otherwise. Because of this, a universally agreed upon set of ethics will never be concretely reached. However, through the actions taken to escape any legal issues, the avoidance of any major ethical issues are avoided.

As no interviews, questionnaires or experiments will take place in this project, other ethical issues regarding this do not need to be considered.

1.4.3 Social and Professional

With smartphones and text messaging services already being a central part of society, this project does not face any social or professional issues.

1.5 Report Structure

The structure of the report from this point onwards will be as follows:

Objectives A discussion of the objectives and goals of the project.

Research Presentation and thorough explanation of all research completed.

Design Full system design, outlining and explaining choices made and tools used.

Development Implementation and software development aspect of the project presented, including the testing phase.

Results Analysis of the developed system accompanied with all conclusions and results found.

Further Work All further work completed for this project and a suggestion and discussion of any improvements that could be made.

Evaluation A conclusion of the project as a whole, including self-assessment.

Acknowledgements A list of acknowledgements relating to this project.

References All reference material used through the project.

Theorems, proofs, code snippets, diagrams and screen shots will be used throughout, as required, to improve understanding and to help explain various sections of this report.

Chapter 2

Objectives

In order to answer the questions described previously, certain goals and objectives must be met. The main objectives of this project where:

1. Research

   - Framework design
   - Currently Available Applications
   - Cryptographic Techniques
   - Relevant factors that can be used to compare schemes implemented on a mobile device

2. Development

   - Framework
     - Design Data Communication framework
       * Server
       * Mobile Application
       * P.C Client
   - Encrypted Data Communication
     - Design and implement cryptographic techniques

3. Analysis

   - Perform tests from research
   - Collect and present results

4. Conclusion

- Present and justify the findings and conclusions that can be made form the completed tests

- Show possible adjustments to the implemented schemes which would increase their usability

5. Further Work

- Detail possible extensions that can be made to the systems to include other possible functions

A further breakdown of the objectives, together with a thorough project task timetable, can be found in the form of a Gantt chart in Appendix A.

Chapter 3

Research

All completed research will now be presented, corresponding with the goals included in the Objectives section of this report.

3.1 Framework

The system framework refers to the developed software that will facilitate the communication of data messages between any two users of the system, whilst not including any form of encryption or decryption. The research required to design a system framework was fairly minimal as most aspects of the software development for this project had already been covered, however, how a full system should be set up had not. As the Java programming language had already been determined as the language that would be used in this project, the finer points of data communication required research. Android application development in this setting, along with how user data should be stored, also required some research.

3.1.1 System setup

The backbone of the developed system was the sending and receiving of data messages between two clients. An initial approach could be that a client, be it a P.C or Android application based client, would send the user input message directly to the recipient. This, however, would required a constant link between all clients of the system, which is not practical due to users having different access habits or varied availability. It also requires a lot more computational power and resources, as each client has to track and directly communicate with each other client, which also is very fragile and prone to errors.

Another approach, the approach used in this project, was found whilst researching using the book by Daniel Liang [10]. The method revolves around the development of a

central server, which can be connected to by clients of the system via sockets [11], also known as a multi-client server. Data messages can then be sent to the server by each client, accompanied by the user ID of the intended recipient of the message. This message is then saved and can be sent to the intended user the next time the user logs on to the system or, if the user is already logged on to the system, the next time they 'refresh', asking the server to send any new messages. This is a slight adaptation to the method described by the reference material, which describes each communication link be created as a separate 'session' but, as with the initial approach, requires both users to be logged on to the system at the same time and simulates a more 'instant' messaging service. The adaptation to the researched method allows users to send and retrieve messages independently and is a more practical approach. This is particularly more practical for smartphone users where connection to the internet or a server can be indeterminate.

Consideration was made as to how more than two users of the system could simultaneously communicate, but it was decided that as this is a feature not present in standard, unencrypted message communication for smartphones, it would not be required for this developed system. Utilising a central server creates a much more robust system, as the server will not be prone to failure (due to the nature of a server) and has the resources to handle almost all of the data management and tasks required for the system to function. It also acts as the central point of the system, allowing for updates to be made without a complete overhaul of the system.

3.1.2   Android Application

With the availability of various resources [12] [13], the research into android application development for this project could be completed efficiently. Although various methods for data communication with android applications existed, it was found that socket-based network programming could be used in the same way as for P.C clients. Other methods involved using HTTP Get and Post commands. Using the same technique for both clients was the logical choice for the development of the framework for this project.

Data compatibility between the clients will be ensured through the use of the Extensible Markup Language (XML) file structure. This file structure can be received and parsed by software developed for both platforms. 'A typical xml document contains sequences of nested tags describing and evaluating a multitude of objects and structures without any constraints, apart from those imposed by basic xml grammar' [14]. The hierarchy of the information contained in an XML document can be accurately and correctly shown through the creation of a conceptual data tree. This allows the XML document to be precisely and efficiently parsed.

An example, taken from w3schools.com [15], of a basic XML document more clearly
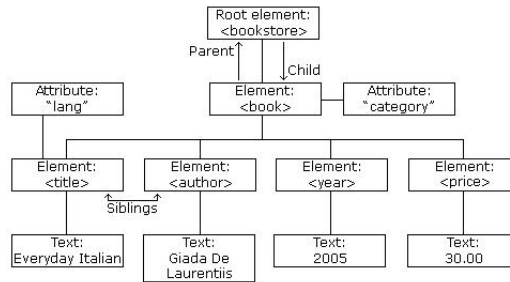
Figure 3.1: Tree structure representing an XML file for a bookstore.

```xml
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="WEB">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

Figure 3.2: The corresponding XML file for a bookstore.

shows the structure and formatting used. Figure 3.1 and Figure 3.2 show that <bookstore> is the root element of the document, and that all <book> elements are contained within <bookstore>. The book element has 4 'children': <title>, <author>, <year> and <price>. These children are used to describe and give properties to the associated book. Within this project the XML files will be set out similarly, with the full document set up displayed in the design section of this report.

3.1.3 Data Storage

In order to store and manage the data for each user (I.D, encryption information, messages), MySQL [16] will be used. MySQL is the most popular and highly used open source relational database management system (RDBMS), with a plethora of documentation and tutorials available. It provides user access to a server which hosts a number of databases. This server can be set up locally on through an external provider. In order to connect, access and edit the database through the developed system, JDBC will be

used. 'JDBC is a set of programming APIs that allows easy connection to a wide range of databases (especially relational databases) through Java programs' [17]. JDBC programming involves creating and executing SQL queries or 'statements' and processing the returned ResultSet objects. Statements contain the commands to select or edit particular entries in the database. A full description of the design of the database system in this project will be included in the following chapter. Because of the points mentioned above there was no reason that MySQL should not be chosen, so an alternative form of database management was not researched.

3.2 Available Applications

The research completed into the applications for encrypted message transfer that are currently available will now be presented. The two 'non-standard' applications were the only applications that could be found on the google play marketplace [5] through the use of various related search keywords. Most applications that where returned focused on encrypting stored data on the smarphone, as opposed to encrypting messages for sending or receiving, showing that there is a requirement in the market for more secure message communication applications. A full explanation and study of the relevant encryption methods will be displayed in the next section.

3.2.1 Short Messaging Service

The short messaging service is a text messaging service featured on all makes and models of mobile and smart phone. The user enters the recipients mobile telephone number and can send a short message (of only text) to the recipient. These messages are encrypted between the phone and base station, however, it is required by law that government enforcement agencies can and should be able to conduct lawful surveillance of SMS messages when required [18]. Because of this, unlawful access to SMS messages is possible, by means other than actually breaking the encryption on the messages.

3.2.2 Cloak SMS

Cloak SMS [19] is an Android application developed by Hamish Medlin. The features of this application are:

- Send and Receive AES Encrypted SMS messages

- Threaded Conversations

- Application lock

- Custom themes

- File Encryption

Figure 3.3 shows the screen used for inputting a message and recipient information. The password is used in the key creation for the encryption method. The main feature of this application, which is the goal of this research to find out, is that it utilises the AES encryption method. Application locking and file encryption are integrated features which are provided by other applications developed by Hamish and whilst useful in practice, are not relevant to this project. Custom themes and threaded conversations are present for marketability and design as opposed to increasing the security of the application. The cost of this application is £1.00, with a free version available with limited features. Overall the application has generally positive feedback, with an average rating of 4.8/5.
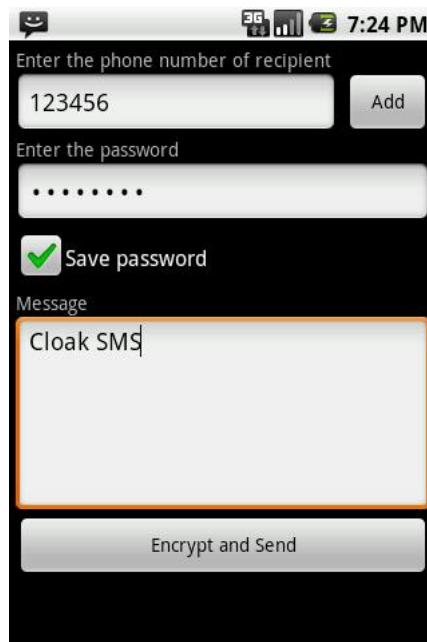
Figure 3.3: A screen capture of the application in use.

3.2.3   RSA Cipher Cat

RSA Cipher Cat is an application developed by Miasoft [20], which makes use of the RSA public-key encryption scheme. Figure 3.4 shows the two main screens of the application, one for encryption (3.4a) and one for decryption (3.4b). The disadvantage of this application is that it very basic, as it does not facilitate the sending of the messages, it only performs the actual encryption and decryption, using files given to the application by the user. It is then up to the user to send the encrypted message however they wish. Whilst it does not have the full features of the other applications, it is perfectly appropriate for
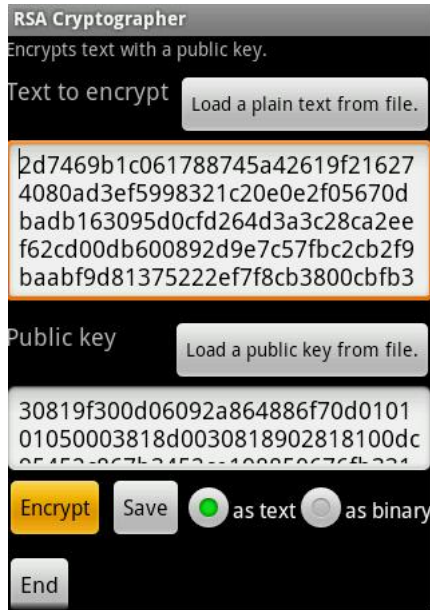
(a) Encrypting a message (b) Decrypting a message
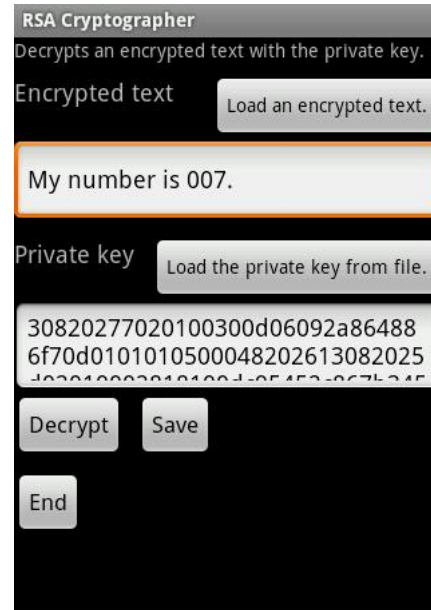
Figure 3.4: Application usage screen captures

this project, as it gives an insight into the current state of encrypted message sending applications and an example of cryptography in practice. The application is free to the public, but has a low user review rating of 1/5, submitted by a single user, whilst having between 500-1,000 installs.

3.3 Cryptographic Techniques

In this section of the report the cryptographic techniques, found as a result of the research completed, will be described and detailed in full, including examples and proofs where necessary. Research was completed using books by Wenbo Mao [21] and William Stallings [22]. Any other texts or materials used within this research section will be individually referenced.

It is important to firstly note that it was found that cryptographic schemes can be loosely sorted into two main categories; symmetric or asymmetric key algorithms.

Symmetric key algorithms feature an encryption key that is identical (or easily computable from) the decryption key. This commonly means that decryption is similar to running the encryption in verse.

Asymmetric key algorithms, on the other hand, operate using two keys called a keypair; a publicly known key (public key) and a key kept in secret (private key) for each user. If user A wishes to send a message to user B, A encrypts the message using B's

17

public key. The public and private key for each user are related in such a way that B can decrypt the message from A using B's private key. However, it is computationally infeasible for A (or any other user) to find B's private key from the corresponding public key.

Another way in which schemes can be categorised is by whether they operate on a block of input data (block ciphers) or a stream of input data (stream ciphers). As the categories are determined by the fundamental workings of the schemes, these categorisations are identical and present in all cryptographic related texts.

3.3.1 Advanced Encryption Standard

The Advanced Encryption Standard (AES) algorithm is a symmetric-key block cipher and was chosen in 1997 by the United States' National Institute of Standards and Technology (NIST) to replace the previously used DES algorithm. AES typically uses a 128-bit block size and a key size of 128, 192 or 256.

3.3.1.1 Finite Field Arithmetic

The internal functions of the AES cipher operate on 8-bit bytes, with the operations of addition, multiplication and division performed over the finite field $GF(2^8)$. Arithmetic operations on integers are found in almost all encryption algorithms. If the arithmetic operation of division is required in the operation, then the arithmetic must be defined over a field, due to the fact that division requires that each non-zero element must have a multiplicative inverse. Furthermore, to increase implementation efficiency, it is required that integers fall into a given number of bits, with no wasted bit patterns; integers in the range 0 to $2^n - 1$, which fit into an n-bit word. However, the set of integers that follow this property, Z_{2^n}, using modular arithmetic, is not a field as, for example, 2 has no multiplicative inverse in the set. This means that there is no integer b, such that $2b$ mod $2^n = 1$. Therefore, the finite field containing 2^n elements, $GF(2^8)$, is used. Taking the set, S, of all polynomials of degree $n-1$ or less with binary coefficients, every member is of the form

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ... + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

where each a_i is either 0 or 1. Therefore, S consists of '2^n different polynomials. For $n = 3$, the elements of S are:

$$\begin{array}{llll} 0 & x & x^2 & x^2 + x \\ 1 & x+1 & x^2+1 & x^2+x+1 \end{array}$$

The ordinary rules of polynomial arithmetic using the basic rules of algebra is followed by arithmetic performed on elements of this set, with two alterations. Firstly, arithmetic performed on the coefficients is performed modulo 2. This is identical to the XOR operation. Secondly, if multiplication results in a polynomial of degree greater than $n-1$, then the polynomial is divided by some irreducible polynomial $m(x)$ of degree n, with the remainder kept (polynomial is reduced modulo $m(x)$). For $f(x)$, the result is $r(x) = f(x)$ mod $m(x)$. $m(x)$ is described as irreducible if and only if $m(x)$ cannot be expressed as a product of two polynomials, both with degree lower than that of $m(x)$. These definitions result in S being a finite field.

Any polynomial in $GF(2^n)$ can be uniquely represented as $(a_{n-1}a_{n-2}...a_0)$, its n binary coefficients. Therefore, every polynomial in $GF(2^n)$ can be represented by an n-bit number.

Taking the bitwise XOR of two terms is the equivalent of addition.

Multiplication of an element in $GF(2^n)$ by 2 is made up of a left shift, followed by a conditional XOR with a constant. Repeated application of this rule can be used to achieve multiplications of higher orders.

For example, AES uses arithmetic in the finite field $GF(2^8)$ with $m(x) = x^8 + x^4 + x^3 + x + 1$. Taking two elements of $GF(2^n)$

$$A = (a_7 a_6 ... a_1 a_0)$$

$$B = (b_7 b_6 ... b_1 b_0)$$

$$A + B = (c_7 c_6 ... c_1 c_0)$$

where $c_i = a_i \oplus b_i$ and

$$\{02\} \bullet A = \begin{cases} (a_6...a_1 a_0 0) & \text{if } a_7 = 0 \\ (a_6...a_1 a_0 0) \oplus (00011011) & \text{if } a_7 = 1 \end{cases}$$

3.3.1.2 Structure

The AES cipher takes an input plaintext in blocks of 128 bits (16 bytes). The key used can have a length of 128, 192 or 256 bits (16,24 or 32 bytes). The input block (and the chosen key) can be depicted as a 4x4 matrix of bytes. This input block is copied into a State array, which is the array that is operated upon during for the duration of the algorithm. The resulting State array is finally copied into an output matrix once the transformations have been completed. The algorithm operates by performing a number of transformations on the input plaintext block. The number of rounds of transformations is determined by the size of the key chosen.

16 byte key - 10 rounds

24 byte key - 12 rounds

32 byte key - 14 rounds

The key chosen for the encryption scheme is initially expanded, using the original 4x4 byte key, to provide a 4x4 byte Round Key for each round. This means that the initial 16 byte key (4 words) is expanded to 176 bytes (44 words). This provides 11, 4 word (4x4 byte) rounds keys. As the original key is 16 bytes, 10 rounds of transformations will be applied, plus an initial usage of a Round Key, hence 11 Round Keys are required.

The structure of the AES cipher is as follows:

1. Round 0

    (a) AddRoundKey(State,RoundKey)

2. Round 1 to N-1

    (a) SubBytes(State)

    (b) ShiftRows(State)

    (c) MixColumns(State)

    (d) AddRoundKey(State,RoundKey)

3. Final Round

    (a) SubBytes(State)

    (b) ShiftRows(State)

    (c) AddRoundKey(State,RoundKey)

Before describing each transformation in detail, some interesting and important points regarding the structure of the AES algorithm shall be made. Firstly, AES does not follow a typical Feistel structure, where half of the data block is used to modify the other half and then the halves are swapped. Instead, the entire data block is processed as a single matrix during each round.

Also, the AddRoundKey stage is the only stage that makes use of the key. Therefore, the cipher must begin and end with an AddRoundKey stage, as any other stage applied before or after this would be reversible without knowledge of the key, which would add no security. The AddRoundKey stage is, by itself, not formidable. The other three stages create confusion, diffusion and non-linearity, enabling the algorithm to be described as

an alternation of the XOR encryption (AddRoundKey) of a block and the scrambling of the block (other three stages). This creates an efficient and highly secure scheme.

Lastly, the final round of encryption (and decryption) consists of only three stages. This is as a result of the structure of AES and is required for the cipher to be reversible.

3.3.1.3 Functions

Each function will now be taken and described individually.

Sub-bytes The sub-bytes transformation consists of a non-linear substitution of each byte of the input State. The following equation (using the rules of arithmetic defined previously) is used to transform the non-zero input byte $x \in \mathrm{GF}(2^8)$:

$$y = Ax^{-1} + b \tag{3.1}$$

where $A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$ and $b = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$

If x is the zero byte, then $y = b$ is the result.

More commonly, the transformation in (3.1) is used to generate a simple lookup table called an S-box, shown in Figure 3.5. The S-box is a permutation of all possible 256 8-bit values. Each byte of the input State can be substituted with the new value by using the leftmost 4 bits of the byte as the row value and the rightmost 4 bits as the column value. Using these values as indexes to the new value from the S-box, resulting in a new 8 bit output value. For example, the hexidecimal value $\{89\}$ references row 8 and column 9 of the S-box and will thus be mapped to $\{A7\}$. The constant in (3.1) has been chosen so that no fixed points exist within the S-box (S-box$(a) \neq a$). Furthermore, no 'opposite' fixed points exist (S-box$(a) \neq \bar{a}$), where $\bar{a}$ is the bitwise complement of a.

The non-linearity of the transformation in (3.1) is a result of the inversion x^{-1} (multiplicative inverse) only. Since the rows of A are linearly independent in $\mathrm{GF}(2^8)$, the transformation (3.1) and hence the function Sub-Bytes, is invertible. The inverse S-box can be generated in the same way using the inverse of (3.1).

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1x	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2x	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3x	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4x	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5x	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6x	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7x	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8x	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9x	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
ax	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
bx	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
cx	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
dx	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
ex	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
fx	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 3.5: AES S-box

ShiftRows   The operation ShiftRows is described as a transposition cipher, only the positions of the elements are rearranged not their identities. For elements in the ith ro $(i = 0123)$, the rearrangment is a cyclic shift by $4 - i$ positions. This is shown in Figure 3.6.
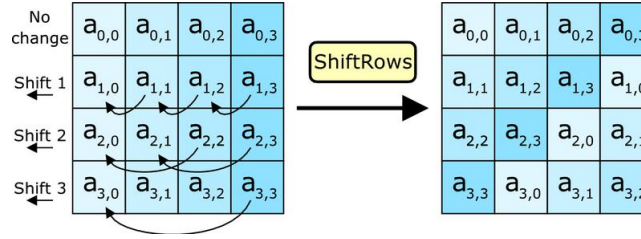
Figure 3.6: ShiftRows transformation

It can be clearly seen that the result of this transformation is that the elements (4 bytes) of each original column are spread out to four different columns in the output, which is the rationale behind this operation. As only the positions are changed or elements in the State, the transformation is clearly invertible.

MixColumns   The MixColumns function operates on each column individually, with each byte of the column mapped into a new value that is determined by the value of all four bytes in that column. The transformation can be defined by a matrix multiplication on the input state shown in Figure 3.7 taken from [23].

All additions and multiplications are performed in $GF(2^8)$. The transformation on a single column of the input State can be seen in Figure 3.8 ([24]).

Figure 3.9 is an example of the MixColumns transformation.

Recalling the rules for arithmetic in $GF(2^8)$ presented previously, the first resulting element of the example can be verified.

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus (\{46\}) \oplus (\{A6\}) = (\{47\})$$

22

Figure 3.7: MixColumns transformation

$$s'_{0,j} = (2 \bullet s_{0,j}) \oplus (3 \bullet s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \bullet s_{1,j}) \oplus (3 \bullet s_{2,j}) \oplus s_{3,j}$$
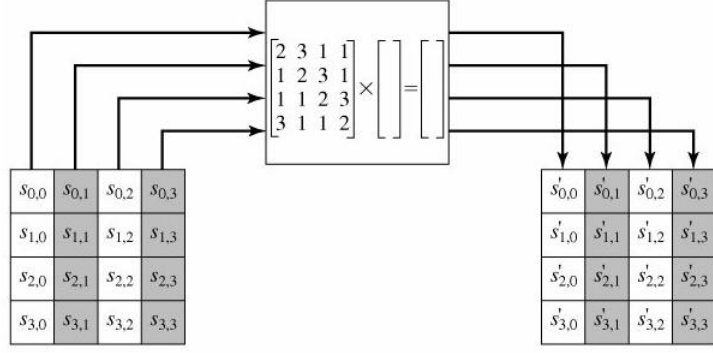$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \bullet s_{2,j}) \oplus (3 \bullet s_{3,j})$$
$$s'_{3,j} = (3 \bullet s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \bullet s_{3,j})$$

Figure 3.8: MixColumns transformation on a single column

Figure 3.9: MixColumns transformation example

$$(\{02\} \bullet \{87\}) = (00001110) \oplus (00011011) = (00010101)$$
$$(\{03\} \bullet \{6E\}) = (\{6E\} \oplus (\{02\} \bullet \{6E\})) = (01101110) \oplus (11011100) = (10110010)$$

Then

$$(\{02\} \bullet \{87\}) \oplus (\{03\} \bullet \{6E\}) \oplus (\{46\}) \oplus (\{A6\})$$

is equal to

$$(00010101) \oplus (10110010) \oplus (01000110) \oplus (110100110) = (01000111) = (\{47\})$$

The other equations can be verified in a similar way. The inverse of this operation can be computed using the matrix multiplication found in Figure 3.10.

The coefficients chosen for the matrix in Figure 3.7 ensure a sufficient mixing among the bytes of each column. A combination of the MixColumns and ShiftRows operations ensures that after a few rounds, all the output bits depend upon all of the input bits.

AddRoundKey The AddRoundKey operation is the final transformation for each round. The 128 bits of the input State are bitwise XORed (addition in $GF(2^8)$) with

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Figure 3.10: The inverse MixColumns operation

the 128 bits of the current round key. The operation is viewed as a columnwise operation between the 4 bytes of the input State and one word of the round key and is therefore a byte-level operation. Figure 3.11 shows an example of the AddRoundKey operation, where the first matrix is the State and the second is the Round Key.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

Figure 3.11: An example of the AddRoundKey operation

The inverse of this operation is identical to this due to the fact that the XOR operation is its own inverse. The AddRoundKey transformation affects every bit of the input State and is as simple as possible. However, security is ensured through the complexity of the round key expansion and the complexity of the other stages of AES.

3.3.1.4 Key Expansion

As mentioned previously, key expansion is required to expand the key into a number of Round Keys, one for each round of the AES algorithm. For example, a four-word (16 byte) key is expanded into an array of 44 words (176 bytes), providing a four-word round key for each of the 10 rounds plus the initial AddRoundKey.

To begin, the four-word key is copied into the first four words of the expanded key. The remaining words of the expanded key are generated and copied four words at a time. Each new word $w[i]$ is calculated depending on the preceding word, $w[i-4]$ and the word four positions before it, $w[i-4]$.

For three of the four cases, the two words are combined using the XOR operation. However, if the words position in the array is a multiple of 4, a more complex function is used. Figure 3.12 shows the key expansion for the first eight words of the expanded key (two round keys). The function g represents the complex function used for a word in a position that is a multiple of 4.

The function g is comprised of the following:

1. **RotWord** One-byte circular left shift on a word. This means input word $[B_0,B_1,B_2,B_3]$ becomes $[B_1,B_2,B_3,B_0]$.

24
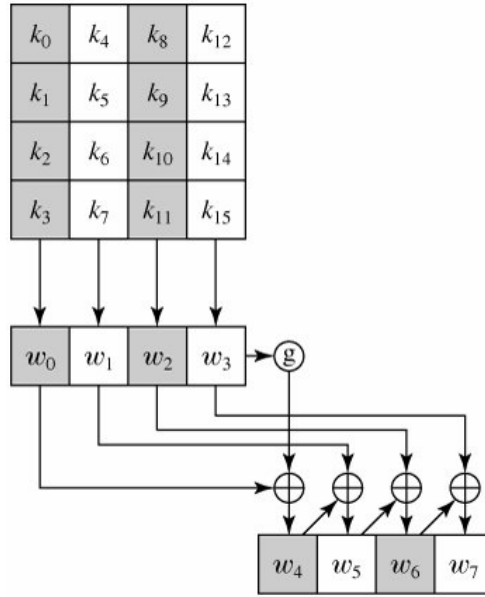
Figure 3.12: Basic key expansion diagram

2. **SubWord** Byte substitution on each byte of the input word, using the S-box (Figure 3.5).

3. **Rcon** The output from steps 1 and 2 are XORed with a round constant, Rcon[j].

The effect of an XOR between the round constant and a word is to only perform an XOR on the left-most byte of the word, due to the fact that the round constantt is a word in which the three right-most bytes are always 0. Each round uses a different round constant and is defined as Rcon[j] = (RC[j],0,0,0) with RC[1] = 1, RC[j] = 2 • RC[j-1], with multiplication defined over the field $GF(2^8)$). In hexidecimal, the values of RC[j] are found in Table 3.1.

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

Table 3.1: Hexidecimal RC[j] values

Figure 3.13 shows a Java based pseudo-code representation of the key expansion algorithm.

The design and inclusion of a round key means that:

• If part of the cipher key or round key is known, then the calculation of other round key bits is not possible.

• Symmetry or similarity between the ways in which round keys are generated in different rounds is eliminated (use of a round-dependent constant)

25

```
KeyExpansion(byte key[16], word w[44])
{
        word = temp;
        for (i = 0; i < 4; i++) {
                w[i] = (key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
        }

        for (i = 4; i <44; i++) {
                temp = w[i-1];
                if (i mod 4 = 0) {
                        temp = XOR(SubWord(RotWord(temp)), Rcon[i/4]);
                }
                w[i] = XOR( w[i-4] , temp)
                }
}
```

Figure 3.13: Key Expansion

- Each key bit affects many of the future round key bits (diffusion of the cipher key).

- Only enough non-linearity is used to ensure that the full determination of round key differences from cipher key differences is prohibited.

3.3.1.5 Decryption

Decryption of the AES cipher algorithm essentially involves running the encryption scheme 'backwards' or applying the encryption scheme again but in reverse, using the inverse operations of those operations used for encryption.

As shown previously, each stage is clearly reversible. SubBytes, ShiftRows and Mix-Columns each have an associated inverse function. With AddRoundKey, the inverse is calculated by XORing the same round key to the block, due to the fact that A $\oplus$ B $\oplus$ B = A. Now that it is established that all four operations are reversible, it can be easily verified that the correct plaintext is recovered through decryption.

Figure 3.14 shows the comparison between encryption and decryption. At each horizontally equal point, the value of the State is the same for both encryption and decryption. In order for the structure of the cipher to be reversible, both the encryption and decryption end with a final round of three stages.

3.3.1.6 Example

Instead of presenting a formal example of the AES cipher, the reader is directed to the example by William Stallings on page 193 of [22]. A formal example is not displayed due to the complexity of computing the calculations required by hand and the fact that an
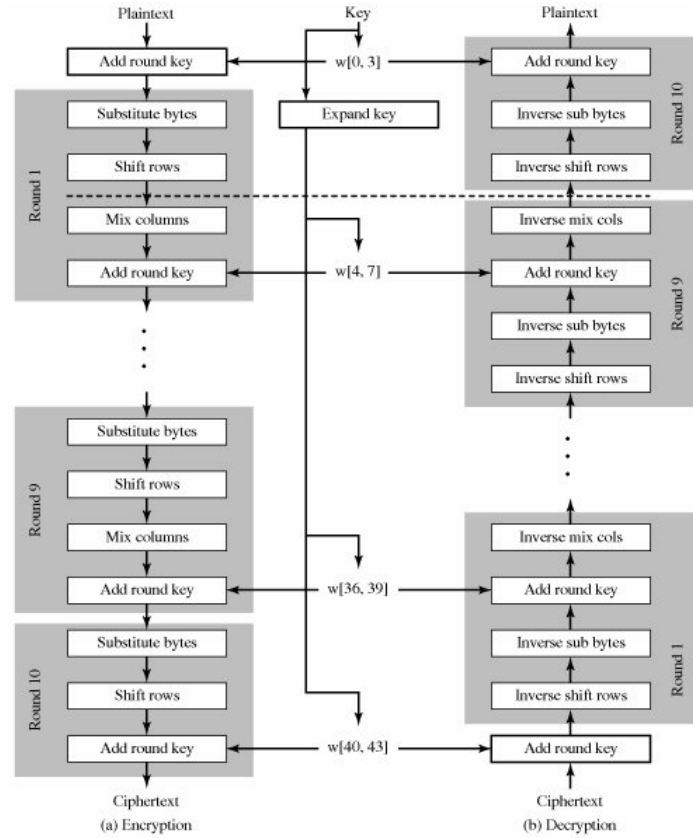
Figure 3.14: AES Encryption and Decryption

example would add no benefit to this section of the report, as an understanding of this scheme has already been shown through the explanation of its operation.

3.3.1.7 Implementation

As encryption schemes are used primarily in computer based systems, consideration must be made about how the functions that make up the cipher are implemented. Different implementations of a similar function made lead to a suboptimal or insecure system. It can be seen from the description of the AES cipher that the internal functions are fairly simple and operate in small algebraic spaces. Because of this, implementations of the operations can be done with extremely good efficiency. AddRoundKey is a bytewise XOR operation and ShiftRows is a simple byte-shifting operation, so can be implemented with good efficiency on any size processor. SubBytes and MixColumns, however, have non-trivial algebraic operations and therefore should be looked at in more detail to find faster implementation methods.

SubBytes, as shown in the description of the operation, can be completed using a table lookup method, rather than calculating individual values each time. The table of $2^8 = 256$ pairs of bytes can be generated once and used forever (hard-coded into either

hardware or software). The table lookup method is not only efficient, but also prevents a timing analysis attack, which is based upon a malicious user observing the operation time difference for different data which may suggest whether an operation is performed on bit 0 or bit 1. Inversion can clearly be completed using the inversion table. Therefore, SubBytes can be implemented using two, 256 byte, tables.

MixColumns can also be improved using a table lookup method. All multiplications (within the field) required to complete MixColumns, are of the form $z = x \cdot y$ where $x \in \{01, 02, 03\}$ and $y \in GF(2^8)$ (see Figure 3.8. As the byte 01 is simply the multiplicative identity in the field $(01 \cdot y = y)$, an implementation of this multiplication table only requires 2 x 256 = 512 entries. Using this implementation technique not only increases operation speed, but also decreases the risk of a timing analysis attack.

3.3.1.8 Summary

To conclude the study of the AES cipher, a short summary:

- The cipher contains a number of rounds of operations, the number of which is determined by the key size.

- Key sizes normally take the values: 128-bits (10 rounds), 192-bits (12 rounds) and 256 (14 rounds).

- The key is expanded to provide a 4-word round key for each round.

- The internal functions of the AES algorithm are:

  - SubBytes: non-linear substitution.

  - ShiftRows and MixColumns: mixture of byte positions of the input State.

  - AddRoundKey: secret randomness to the message distribution.

3.3.2 Rivest-Shamir-Aldeman Algorithm

Published in 1978, the Rivest-Shamir-Aldeman (RSA) scheme [25] is one of the most widely accepted and implemented approaches to public-key encryption. The RSA algorithm is based upon the difficulty of factoring large integers. It acts as a block cipher in which for some n, the ciphertext and plaintext are integers between 0 and n. Typically, n is less than 1024 bits or 309 decimal digits (n is less than 2^{1024}). Hence, the actual message is split into plaintext blocks using a chosen encoding method, with each block encrypted and decrypted separately.

The operation of the RSA encryption scheme has three parts: Key generation, Encryption and Decryption. The description of this scheme uses the common placeholder names of Alice and Bob inkeeping with tradition set out by Bruce Schneier [26].

3.3.2.1 Background

The RSA algorithm utilises mathematical techniques and results which will be presented now, before the main algorithm is described.

Euler's Totient Function Written as $\phi(n)$, Euler's totient function is defined as the number of positive integers less than n and relatively prime (commonly divisible by only 1) to n. By convention, $\phi(1) = 1$.

Clearly, for a prime number p, $\phi(p) = p - 1$, as all positive integers from 1 to $p - 1$ are relatively prime to p. Taking two prime numbers p and q with $p \neq q$, it can be shown that for $n = pq$

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p - 1) \times (q - 1)$$

To show that $\phi(n) = \phi(p) \times \phi(q)$, consider the set of positive integers less than n, $\{1, ..., (pq - 1)\}$. The integers that are not relatively prime to n in this set are the sets $\{p, 2p, ..., (q - 1)p\}$ and $\{q, 2q, ..., (p - 1)q\}$ (containing $(q - 1)$ and $(p - 1)$ elements respectively). Therefore:

$$\begin{aligned}
\phi(n) &= (pq - 1) - [(q - 1) + (p - 1)] \\
&= pq - (p + q) + 1 \\
&= (p - 1) \times (q - 1) \\
&= \phi(p) \times \phi(q)
\end{aligned}$$

As an example:

$$\phi(21) = \phi(3) \times \phi(7) = (3 - 1) \times (7 - 1) = 2 \times 6 = 12$$

where the 12 integers are $\{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$.

3.3.2.2 Description

The RSA algorithm is an example of an asymmetric key algorithm, in which a public key and a private key are utilised. The public key is shared and known with everyone and used in the encryption process. The messages encrypted with the public key can only be decrypted using the matching private key. For some plaintext block M and ciphertext block C, encryption and decryption are of the following form:

$$C = M^e \bmod (n)$$

$$M = C^d \bmod (n) = (M^e)^d \bmod (n) = M^{ed} \bmod (n)$$

The keys for this algorithm are therefore; the private key PR $= \{d, n\}$ and the public key PU $= \{e, n\}$. For this algorithm to be securely used for public-key encryption, the following requirements must be satisfied.

1. Values of e, d and n can be found such that $M^{ed} \bmod (n) = M$ for all $M < n$.

2. $M^e \bmod (n)$ and $C^d \bmod (n)$ are relatively easy to compute, for all values of $M < n$.

3. It is infeasible to determine d given e and n.

The first required will be considered now, with the other requirements being described later in this section. The relationship that needs to be satisfied is $M^{ed} \bmod (n) = M$, which holds if e and d are multiplicative inverses modulo $\phi(n)$, where $\phi(n)$ is the Euler totient function as described previously. The relationship between e and d can be expressed as:

$$ed \bmod \phi(n) = 1$$

which is equivalent to

$$ed = 1 \bmod \phi(n)$$

$$d = e^{-1} \bmod \phi(n)$$

Therefore, e and d are multiplicative inverses $\bmod \phi(n)$, which is only true if d and e are relatively prime to $\phi(n)$. This satisfies the first requirement for the RSA algorithm (a proof of which will be presented towards the end of this section). The stages of the RSA algorithm can no be presented.

Key Generation   The following process is used to generate the required keys:

1. Choose two distinct prime numbers p and q.

    - These are chosen at random and kept private.

2. Calculate $n = pq$.

3. Calculate $\phi(n) = (p - 1)(q - 1)$

4. Choose e such that $1 < e < \phi(n)$ and e and $\phi(n)$ are coprime

5. Determine d as $d = e^{-1} \bmod \phi(n)$. Here d is the multiplicative inverse of e mod $\phi(n)$.

Therefore the private key consists of $[d, n]$ and the public key consists of $[e, n]$. The private key must be kept secret whereas the public key can be shared. p, q and $\phi(n)$ must also be kept secret as they were used to calculate d. This is how RSA can be described as being based on the difficulty of factoring large numbers. If it where computationally easy to find the prime factors of a number, in this case n, then using the factors p and q, $\phi(n)$ could be easily calculated. Combining this with the knowledge of the public value e would allow the calculation of d, breaking the encryption scheme. Therefore, until a computationally efficient way of finding the prime factors of a large number is discovered, the RSA algorithm remains secure.

Encryption   Bob wants to send a message M to Alice. Alice sends Bob her public key $K_a = e, n$ and keeps her private key secret. Bob computes the ciphertext C corresponding to

$$C = M^e \bmod (n)$$

Bob then sends C to Alice.

Decryption   Alice receives ciphertext C from Bob. She can recover the message M from C using her private key by computing

$$M = C^d \bmod (n)$$

Note: M is an integer between 0 and n, so Alice and Bob must agree upon a method to turn their message text into plaintext blocks (integers) and back again, for example a = 00, A = 26 etc.

3.3.2.3   Example

To aid in the explanation of this algorithm, an original basic example will be presented:

1. Select two prime numbers, $p = 13$ and $q = 11$.

2. Calculate $n = pq = 13 \times 11 = 143$.

3. Calculate $\phi(n) = (p-1)(q-1) = 12 \times 10 = 120$.

4. Select e, relatively prime to $\phi(n) = 120$ and less than $\phi(n)$; $e = 7$.

5. Determine d such that $de \equiv 1 \bmod (120)$ and $d < 160$. $d = 103$ as $103 \times 7 = 721 = (6 \times 120) + 1$.

This results in PU $= \{7, 143\}$ and PR $= \{103, 143\}$. With a plaintext input of $M = 44$, encryption (and decryption) are as follows:

$$C = 44^7 \bmod (143) = 99$$

$$M = 99^{103} \bmod (143) = 44$$

3.3.2.4 Implementation

As with the AES cipher, the computational aspects of the RSA algorithm need to be considered. The main three issues that require consideration are presented bellow.

Exponentiation in Modular Arithmetic As both encryption and decryption require raising an integer to an integer power $\bmod(n)$, processes in which to improve the implemented efficiency should be developed. If the exponentiation is calculated and the result is then reduced modulo n, the intermediate values would be colossal. However, a property of modular arithmetic can be utilised:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

which makes calculations much more practical.

The efficiency of the exponentiation should also be considered. This is because in RSA, the exponents are potentially very large. As an example to show how efficiency can be increased, consider x^8, which can be computed as;

$$x^8 = x \times x \times x \times x \times x \times x \times x \times x$$

or by repeatedly taking the square of the previous partial result forming (x^2, x^4, x^8), a less computationally expensive calculation. This can be extended to calculations like x^{11}, where $x^{11} = x^{1+2+8}$. In this case

$$x^{11} = [(x \bmod n) \times (x^2 \bmod n) \times (x^8 \bmod n)] \bmod n$$

Public Key Operation Through a specific choice of e, the speed of operation of the RSA algorithm utilising the public key can be greatly increased. Most commonly, values for e are chosen as $65537(2^{16} + 1), 3$ or 17. This is due to the fact that each of the three choices contain only two 1 bits, which minimises the number of multiplications required to perform the exponentiation. Very low values of e ($e = 3$), however, decrease the security of RSA, so larger values, primarily 65537, are the most popular.

Key Generation   Because $n = pq$ will be public known to any adversary, the prime numbers p and q must be sufficiently large to prevent their discovery by exhaustive methods. On the other hand, the method used for generating these large numbers must be reasonably efficient. Unfortunately, at present, there are no useful methods for generating arbitrarily large prime numbers, so other techniques must be used to tackle the problem. One procedure is to pick a random odd number of the desired magnitude, tests its primality and repeat if it is not prime. Whilst this may seem to increase the inefficiency of the algorithm, it only needs to be completed relatively infrequently, when two users wish to communicate for the first time. Therefore, an vast increase in security is chosen over an initial, slightly large, time cost. However, using the prime number theorem from number theory[1] and the fact that all even numbers can be immediately rejected, the number of tests required would be of the order of $\ln(N)$. For example, if a prime on the order of magnitude of 2^{200} is required, then about $\ln(2^{200})/2 = 70$ tests would be needed to find a prime.

3.3.2.5   Proof of the RSA Algorithm

The main elements of the RSA algorithm can be expressed in the following equation:

$$M^{ed} \bmod n = M$$

which holds if e and d are multiplicative inverses modulo $\phi(n)$. The relationship between e and d can be expressed as:

$$ed \bmod \phi(n) = 1$$

or that there is an integer k such that

$$ed = k\phi(n) + 1$$

Thus, it will be shown in this section that the following holds

$$M^{k\phi(n)+1} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = M \tag{3.2}$$

Three other results that are required for the proof of Equation 3.2 are

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

$$[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$$

---

[1]The number of primes near N are spaced on average one every ($\ln N$) integers.

and if integers a and n are relatively prime, then

$$a^{\phi(n)} \bmod n = 1$$

This last result is named Euler's theorem.

Proof First we show that $M^{k(p-1)(q-1)+1} \bmod p = M \bmod p$. Here, there are two cases to consider:

M **and** p **are not relatively prime** (p divides M) This means that $M \bmod p = 0$ and $M^{k(p-1)(q-1)+1} \bmod p = 0$ and so they are equal.

M **and** p **are relatively prime** By Euler's theorem $M^{\phi(p)} \bmod p = 1$.

It follows that

$$
\begin{aligned}
M^{k(p-1)(q-1)+1} \bmod p &= M[M^{k(p-1)(q-1)1}] \bmod p \\
&= M[(M^{(p-1)})^{k(q-1)}] \\
&= M[(M^{\phi(p)})^{k(q-1)}] \\
&= (M \bmod p) \times [(M^{\phi(p)}) \bmod p]^{k(q-1)} \\
&= (M \bmod p) \times (1)^{k(q-1)} \\
&= (M \bmod p)
\end{aligned}
$$

Which shows that

$$[M^{k(p-1)(q-1)+1} - M] \bmod p = [M^{k(p-1)(q-1)+1} \bmod p] - [M \bmod p] = 0$$

Thus, p divides $[M^{k(p-1)(q-1)+1} - M]$. The same reasoning can be used to show the q divides $[M^{k(p-1)(q-1)+1} - M]$. Because p and q are distinct primes, there must exist an integer r that satisfies

$$[M^{k(p-1)(q-1)+1} - M] = (pq)r = nr$$

Therefore, n divides $[M^{k(p-1)(q-1)+1} - M]$, and so

$$M^{k\phi(n)+1} \bmod n = M^{k(p-1)(q-1)+1} \bmod n = M$$

3.3.2.6   Summary

The RSA algorithm is a public-key scheme, based upon the difficulty of factoring large numbers. Figure 3.15 summarises the operation of the algorithm.

Key Generation	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$
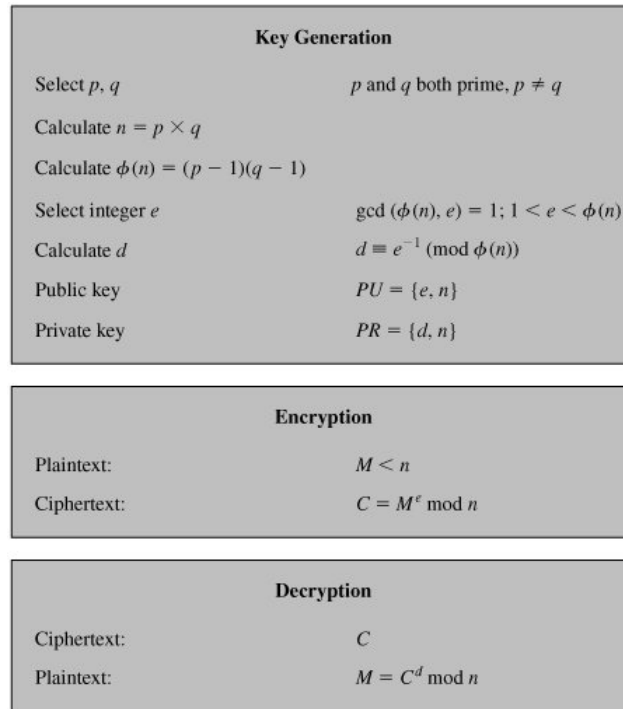
Figure 3.15: Summary of the RSA algorithm

3.3.3 Elliptic Curve Cryptography

3.3.3.1 Introduction

Elliptic Curve Cryptography (ECC) is another approach to public-key cryptography (asymmetric key algorithm) but is based on the algebraic structure of elliptic curves over finite fields. The assumption that finding the discrete logarithm of a random elliptic curve element, with respect to a publicly known base point, is infeasible is the basis for the elliptic curve cryptographic scheme.

As with RSA, the ECC scheme is made up of three parts; key generation, encryption and decryption.

3.3.3.2 Background

This section will provide the background concepts and mathematical foundations on which ECC is based.

Abelian Groups   An abelian group G, is a set of elements with a binary operation $\bullet$, that associates to each ordered pair (a, b) of elements in G an element $(a \bullet b)$ in G, such that the following axioms are obeyed;

(A1) Closure: If $a, b \in G$ then $a \bullet b \in G$.

35

(A2) Associative: $a \bullet (b \bullet c) = (a \bullet b) \bullet c \ \forall a, b, c \in G$.

(A3) Identity Element: $\exists e \in G$ such that $a \bullet e = e \bullet a = a \ \ \forall a \in G$

(A4) Inverse Element: For each $a \in G \ \exists a' \in G$ such that $a \bullet a' = a' \bullet a = e$.

(A5) Commutative: $a \bullet b = b \bullet a \ \forall a, b \in G$.

Note that the operator $\bullet$ is generic and can reger to an mathematical operation.

Elliptic Curves over Real Numbers Elliptic curves are named as such, not becase they are ellipses, but because they are described by cubic equations, similar to those used for calculating ellipse circumferences. The purposes of cryptograpy, elliptic curves are cubic equations of the form:

$$y^2 = x^3 + ax + b \tag{3.3}$$

Included in the description of an elliptic curve is the 'point at infinity' or the 'zero point' denoted O. The equation

$$y = \sqrt{x^3 + ax + b}$$

is used to plot such a curve, which consists of a positive and negative value of y for each x value and so is symmetric about $y = 0$.

Consider the set of points $E(a, b)$ consisting of all the points (x, y) that satisfy Equation 3.3 together with element O. Varying the pair (a, b) results in a different set $E(a, b)$ and thus a different elliptic curve. Figure 3.16 shows two examples of elliptic curves.

A group can be defined based on the set $E(a, b)$ for specific values of a and b in Equation 3.3, provided the condition

$$4a^3 + 27b^2 \neq 0 \tag{3.4}$$

is met. To define the group, the operation called addition (denoted by $+$), for the set $E(a, b)$ (where a and b satisfy Equation 3.4), needs to be defined. In geometric terms, the rules for addition can be stated as: If three points on an elliptic curve lie on a straight line, their sum is O. From this, the rules of addition over an elliptic curve can be defined.

1. The identity element is O, thus $O = -O$ and $\forall P$ on the curve $P + O = P$. The following assumes $P \neq O$ and $Q \neq O$.

2. If $P = (x, y)$ then $-P = (x, -y)$. Thus $P + (-P) = P - P = O$
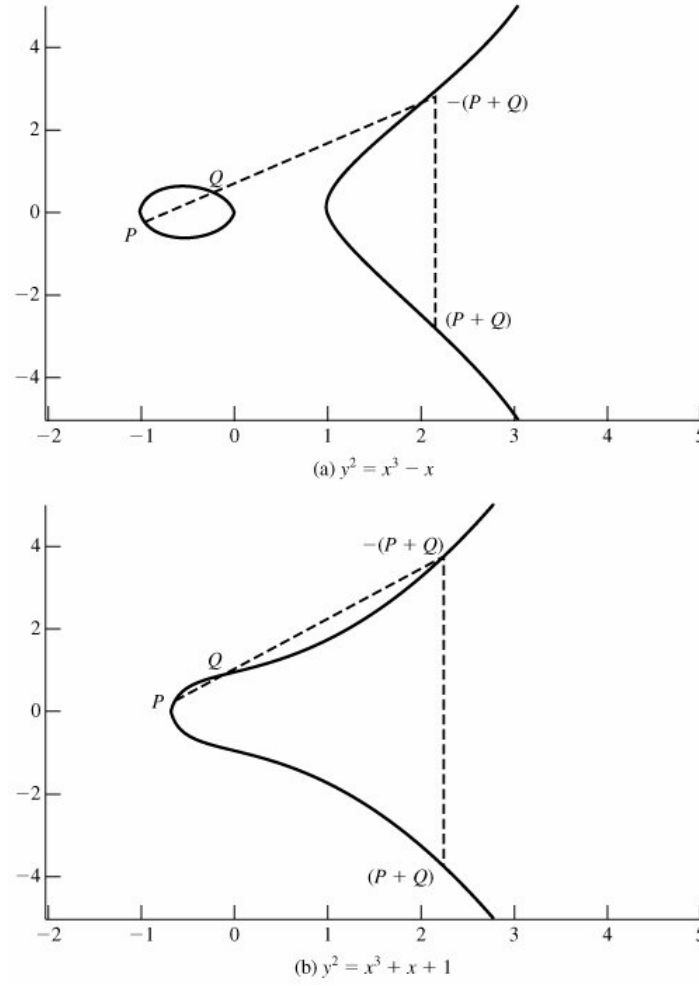
Figure 3.16: Examples of Elliptic Curves

3. The addition of two points P and Q, with different x coordinates, is found on the plotted curve by drawing a straight line between them and finding the third point of intersection R. If the line is a tangent to the curve at P or Q then $R = P$ or $R = Q$. To form a group structure, addition needs to be defined on these three points: $P + Q = -R$, where $P + Q$ is the mirror image (with resepect to the x axis) of the third point of intersection, shown in Figure 3.16.

4. Geometrically, the preceding item also applies to the points P and $-P$, showing that $P + (-P) = O$, which is consistent with item (2).

5. Drawing the tangent line at point Q and finding the other point of intersection S, allows the calculation of $2Q$, where $Q + Q = 2Q = -S$.

With these set of rules, $E(a, b)$ is an abelian group.

The calculation of additions over elliptic curves can now be presented. For two distinct points, $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, where $P \neq -Q$, the slope of the line l that joins

them is $\Delta = (y_Q - y_P)/(x_Q - x_P)$. There is exactly one other point where l intersects the elliptic curve; $-(P + Q)$. After some algebraic manipulation, the sum $R = P + Q$ can be expressed as:

$$x_R = \Delta^2 - x_P - x_Q \tag{3.5}$$

$$y_R = -y_P + \Delta(x_P - x_R) \tag{3.6}$$

A point also needs to be added to itself: $P + P = 2P = R$. When $y_P \neq 0$, the expressions area

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P \tag{3.7}$$

$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P \tag{3.8}$$

Elliptic Curves over Z_p Elliptic curve cryptography utilises elliptic curves in which the variables and coefficients are all restricted to elements of a finite field. Two families of elliptic curves can be used in cryptographic applications: binary curves over $\mathrm{GF}(2^m)$ and prime curves over Z_p. This project focuses on prime curves, which uses cubic equations in which the variables and coefficients all take on values in the set of integers from 0 through $p - 1$ and in which calculations are performed modulo p. Prime curves are best for software applications, due to the fact that extended bit-fiddling operations, that are needed by binary curves, are not required.

The algebraic and geometric interpretation of elliptic curve arithmetic over real numbers does not readily carry over, so a new approach is formed. Firstly, the elliptic curve equation for real numbers (Equation 3.3) is used, but with the coefficients and variables limited to Z_p:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p \tag{3.9}$$

This can be expresses as the set $E_p(a, b)$, consisting of all pairs of integer (x,y) that satisfy equation 3.9, together with a point at infinity O. The coefficients a and b and the variables x and y are all elements over Z_p. As an example, Figure 3.17 shows the points that satisfy $E_{23}(1, 1)$.

A finite abelian group can now be defined based on the set $E_p(a, b)$ provided that $(x^3 + ax + b) \bmod p)$ has no repeated factors. This is equivalent to the condition:

$$(4a^3 + 27b^2) \bmod p \neq 0 \bmod p \tag{3.10}$$
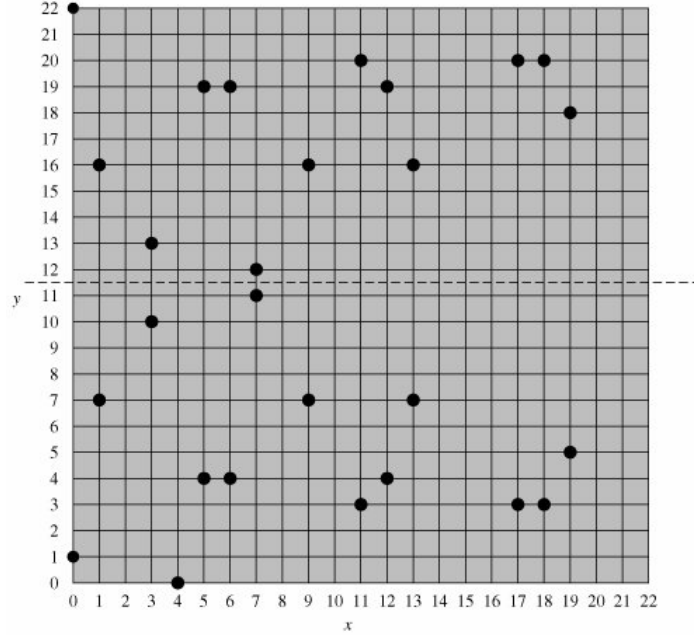
Figure 3.17: Prime Elliptic Curve example

which is of the same form as Equation 3.4.

The rules for addition over $E_p(a, b)$ correspond to the algebraic technique described for elliptic curves defined over real numbers. For all points $P, Q \in E_p(a, b)$:

1. $P + O + P$

2. If $P = (x_P, y_P)$ then $P + (x_P, -y_P) = O$ where $(x_P, -y_P) = -P$

3. If $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$ with $P \neq Q$, then $R = P + Q = (x_R, y_R$ is calculated using:

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$

$$y_R = (\lambda(x_P - x_R) - y_P) \bmod p$$

where

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \bmod p & \text{if } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \bmod p & \text{if } P = Q \end{cases}$$

4. Multiplication is defined as repeat addition; for example $5P = P + P + P + P + P$

3.3.3.3 Description

The following comparisons can be made between RSA and ECC, shown in Table 3.2.

Therefore, to create a cryptographic system using elliptic curves, a problem corresponding to the factoring of two large primes needs to be found. Considering the equation $Q = kP$ where $Q, P \in E_P(a, b)$ and $k < p$, it can be seen that it is relatively easy to

RSA	ECC
Modular multiplication	Addition operation
Modular exponentiation	Multiple addition

Table 3.2: Comparison between RSA and ECC

calculate Q given k and P, but it is relatively hard to determine k given Q and P. This problem is called the discrete logarithm problem for elliptic curves.

To illustrate this, take the group $E_{23}(9, 17)$ which is defined by the equation

$$y^2 \bmod 23 = (x^3 + 9x + 17) \bmod 23$$

Taking the following values of P and Q, the basis of the cipher with these parameters is; what is the discrete logarithm k of $Q = (4, 5)$ to the base $P = (16, 5)$? A brute-force method to calculating k would be to compute multiples of P until Q is found;

$$P = (16, 5), 2P = (20, 20), 3P = (14, 14),$$

$$4P = (19, 20)...8P = (12, 17), 9P = (4, 5)$$

which shows $k = 9$.

In practice, k would be so large, a brute-force approach would be computationally infeasible.

Key Generation   Firstly, a prime integer q and elliptic curve parameters a and b for Equation 3.9 are selected. This defines the elliptic group of points $E_q(a, b)$. A base point $G = (x_1, y_1) \in E_q(a, b)$ whose order is a very large value n, where order n of G is the smallest positive integer n such that $nG = 0$. G and n are global public elements, known to all participants.

User A creates their private key by selecting an integer n_A, less than n. They can then generate their public key $P_A = n_A \times G$. This public key is a point in $E_q(a, b)$. User B creates their private (n_B) and public ($P_B = n_B \times G$) keys in the same way.

Encryption   Encryption begins by encoding the plaintext message m as an $x - y$ point P_m. As not all coordinates are in $E_q(a, b)$, the message cannot just be simply encoded into an $x - y$ point. The point P_m will be encrypted as ciphertext and sent to the recipient to be decrypted.

For user A to encrypt and send the message P_m to user B, A chooses a random positive integer k and uses B's public key P_B to produce the ciphertext C_m as a pair of points:

$$C_m = \{kG, P_m + kP_B\}$$

Decryption To decrypt the ciphertext C_m above, B multiplies the first point in the pair by their private key and subtracts the result from the second point:

$$[P_m + kP_B] - [n_B(kG)] = P_m + k(n_BG) - n_B(kG) = P_m$$

The message P_m has been obscured by A through the addition of kP_B. Only A knows the value of K, so even though P_B is public, no-one can remove the kP_B. If someone knows the private key n_B, however, the 'clue' kG (included by A) can be utilised. For an attacker to recover the message, they would have to compute k given G and kG, which is assumed to be (very) hard.

Now that B knows the point P_m, they can decode the message using the already agreed upon coding method to retrieve the plaintext message m.

3.3.3.4 Example

A basic example will now be completed to demonstrate the encryption process.

Firstly, take

$$p = 751 \text{ and } E_P(-1, 188)$$

which is equivalent to the curve

$$y^2 = x^3 - x + 188$$

Also take $G = (0, 37)$.

For A to send a message to B, A first encodes the message into the elliptic point $P_m = (562, 201)$ and selects the random number $k = 386$. As B's public key is $P_B = (201, 5)$, the process continues as follows:

$$kG = 386(0, 376) = (676, 558)$$

$$P_m + kP_B = (562, 201) + 386(201, 5) = (385, 328)$$

Thus, A sends the ciphertext

$$C_m = \{(676, 558), (385, 328)\}$$

3.3.3.5 Summary

Elliptic Curve Cryptography is an asymmetric encryption scheme which operates with a global public elements $E_q(a, b)$ (elliptic curve group) and G (point on elliptic curve with large order n

The keys used for this scheme are

$$PR = \{n_B\}$$

$$PU = \{P_B = n_B \times G\}$$

and similarly for user B.

Encryption is completed to generate the cipher text C_m by:

$$C_m = \{kG, P_m + kP_B\}$$

with decryption taking place as follows

$$[P_m + kP_B] - [n_B(kG)] = P_m + k(n_B G) - n_B(kG) = P_m$$

An encoding is required to transform the input plaintext message m into a point P_m in the established elliptic group.

3.4 Comparison Factors

As the third section of this project is centred around the analysis and comparison of the various cryptographic schemes implemented through the software development, research was performed in order to discover which factors contribute to how successful or useful the implemented schemes are. Most of the discovered factors were found in the article by Nirav Jobanputra, *et al* [27]. This article studies not only the comparisons of various cryptographic techniques when implemented on a smartphone, but also Bio-information based security solutions such as finger-printing or voice recognition. It should be noted that, as the article was published in 2009, the conclusions have lost some relevance due to the multitude of advancements that have been made regarding smartphones.

That comparison factors that were found that could be used when analysing and concluding the results of this project were:

- **Difficulty of techniques required to break encryption** - Level of various cryptanalytic techniques required to break the encryption.

- **Battery usage** - Percent of battery usage for the duration of the application completing its required tasks.

- **Key generation time** - The time it takes to generate the particular key.

- **Encryption and Decryption time** - The time it takes from start to finish to encrypt and decrypt a particular controlled file.

- **Cryptographic key size** - Size of the key required for the particular cryptographic technique.

- **Data usage** - Amount of data that is stored by the application and the amount of data that is sent or received.

- **Application size** - Size of the application once it has been installed on the smartphone.

- **Number of server connections** - The number of times any sort of data needs to be sent from the server to the phone or the converse.

A combination of these comparison factors will be used to fully compare and contrast all aspects of the cryptographic schemes that will be implemented in this project.

3.5 Conclusion

The results of the research completed for this project has now been presented in full. Research material was mostly found through the Warwick University library and online database, in the form of textbooks, e-books and online papers or articles. As mentioned, the Google play marketplace was used to research the various Android applications. Keywords such as 'Cryptography' and 'Encryption' where used to search the databases, as well as many others relating to the field of study in which research was being completed. The outcome of the research forms the backbone of the project and is of such a quality that the project can continue from this point very successfully. This is due to the depth and breadth of information and knowledge gained from the research as a result of large and thorough research material databases and stores. The materials used are from reliable and prominent authors and the information was consistently found in other texts or forms when cross-checking and verifying the material. Chapters or sections of use were found directly, as opposed to the reading of the whole material from start to finish being required. This is due to the fact that the topic and required outcomes of each section of research was known before hand and allowed the next section of research to be more focused.

As a result of this research, a full system can be designed and implemented to allow for the generation of the results required to answer the main questions and problems specified in this project and to reach specific conclusions and outcomes.

Chapter 4

Design

4.1 Introduction

Within this section a full system overview will be given, accompanied by a more in-depth description of each of the main parts of the system. This will include the; Server, Database, Clients (P.C and Android) and the key sharing protocol. Diagrams will aid in the description of the designed system and will be of the form of Unified Modeling Language (UML) diagrans. UML diagrams are based upon a set of graphic notation techniques used to demonstrate visually object-orientated software systems, found, particularly, described in the book by Minh Duc Bui [28].

4.2 Objectives

The objectives of the design phase of this project can be found in the main 'Objectives' section of this report. However, to reiterate the development specific objectives in an alternate form and in light of the research completed;

1. Develop a data communication framework, which includes;

    (a) Server

    (b) Database

    (c) Mobile Application

    (d) P.C Client

2. Design and implement cryptographic techniques to allow for testing;

    (a) Key exchange protocol

(b) RSA

(c) AES

How these goals will be achieved will be covered throughout this section.

4.3 Main System

4.3.1 Overview

Put simply, the system to be created is a multi-client server facilitating the communication of data between two clients. The data is encrypted (and decrypted) using different cryptographic schemes. Figure 4.1 shows a basic design of the system and how the main sections will interact. The server is the central controlling program (as opposed to a hardware system) communicating with each other piece of the overall system; the database to save, edit or retrieve user data and the clients to send and transfer data messages to other clients of the system.
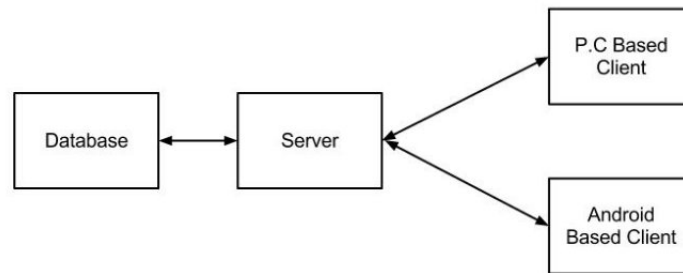
Figure 4.1: Basic System Design

In order to fully explain the system, each system module will now be taken and explained in detail individually.

4.3.2 Server

The primary purpose of the server is to facilitate the sending of messages between any two users of the system. A connection is formed between a client and the server via sockets, with an independent socket connection for each client that is currently communicating with the server. To connect to the server, the client specifies the I.P address of the server and the socket number. The server has full access to the accompanying database, with permission to query or edit database entries. Connection by the server to the database is managed through a database management class.

The server has the ability to save required files in it's own associated disk space. The files saved by the server will be:

- Message response files e.g. 'no new message' files.

- Transmission acknowledgement files.

- Encrypted messages to be sent to clients.

- Cryptographic keys belonging to the server and the clients that have completed the exchange protocol.

Files received from the users of the system will be named using the following template, before being saved, to create a structure file system which will allow for the quick and simple finding of any required files.

to/from ID Day Month Time(hh mm ss GMT) year.xml
e.g. from user8 Mon Mar 04 10 48 GMT 2013.xml

Once a socket connection is made between the server and client, the client user is required to 'sign in' to verify that they have a user account set up with the system already. Once the user has signed in, any messages that the server has waiting to be sent to the current user can then be sent. The 'signing in' protocol, from the point of view of the server, is shown in Algorithm 1.

This protocol fully satisfies the needs of this project, however, a few alterations would need to be made if the system were to be released to the public. To ensure user account security, a log in password should be required and requested of the user when their account is set up and each time they wish to log in. A more unique form of separate identification should be used to reinforce the individual identification number, for example their verified email address, as opposed to the I.P address used above. However, these points do not pose an issue for this project, as the system is created in order to answer specific questions, not as a product to release to the public.

Another important feature of the server is that it acts as a trusted third party within the cryptographic process. It has its own associated public and private keys so that encrypted messages between the server and the client can be sent. If only the clients key was used, encrypted messages could only be sent from the server to the client and not from the client to the server. The importance of this feature will become more apparent in the key exchange protocol description.

Algorithm 1 Client log in protocol
___
Is the client a user?

if client is a system user **then**

    What is the users ID?

    if ID exists in database **then**

        User logged into system

    else

        ID doesn't exist - restart protocol

    end if

else

    if User IP address matches a user database entry **then**

        Client is a system user

        Log in that user ID

    else

        Create a new user entry in the database

        Log in the new user

    end if

end if

Send to the user any waiting messages found in the database

Receive recipient user ID and message from logged in user and save to database
___

4.3.3   Database

The database for this project is built using the MySQL database management system, as described previously, hosted on the 'local host'. This enables the database to be used with the local server, without having to host both the database and server on an external host, which incurs sizeable costs as no completely free hosting exists.

The design of the database is to store, for each user entry, the following information:

- **Unique ID** An ID assigned to each user which is individual and different to all other users

- **I.P Address** The I.P address of the user

- **Pubic Key location** The disk location of the users public key, stored and used by the server

- **Message Location** The storage location of the next message waiting to be sent to the user

- **Key Exchange file locations** Locations of the files used for the key exchange protocol

The Unique ID acts as the primary key, which is used to uniquely identify each record in the database table. The ID is automatically generated by the database management system when a new user is added to the database, escaping the possibility of human error, and consists of a positive integer. The assignment of the primary key is based on an incremented counter, so if an entry is deleted from the table, that user ID will not be used again.

To expand this database to allow for an increased user base another relational database table would be created to store all of the messages in the system, accompanied with the User ID of the recipient of that message. When a user logs into the system, the server would query the database and return all of the messages from the appropriate database intended for that user. However, this feature is not required for this project, but is worth mentioning in preparation for the 'Further Work' section.

As mentioned previously, the database is managed by the server through the use of a database management class. This class is responsible for connecting to the hosted database and querying data entries, among other tasks. The class diagram for this module can be seen in Figure 4.2

```
                        DBManager

connection : Connection
driverName : String = "com.mysql.jdbc.Driver"
serverName : String = "localhost"
portNumber : String = "3306"
sid : String = "typdb"
url : String = "jdbc:mysql://" + serverName + ":" +
portNumber + "/" + sid;

username : String = "TYPuser"
password : String = "TYPpass"

doConnection ( ) : Boolean
returnIPfromID ( id : int ) : String
returnFirstMessagefromID ( id : int, loc : int ) : String
userIDExists ( id : int ) : Boolean
userIPExists ( ip : String ) : Boolean
addNewUser ( ip : String ) :int
updateUser ( userid : int, field : String, data : String )
```
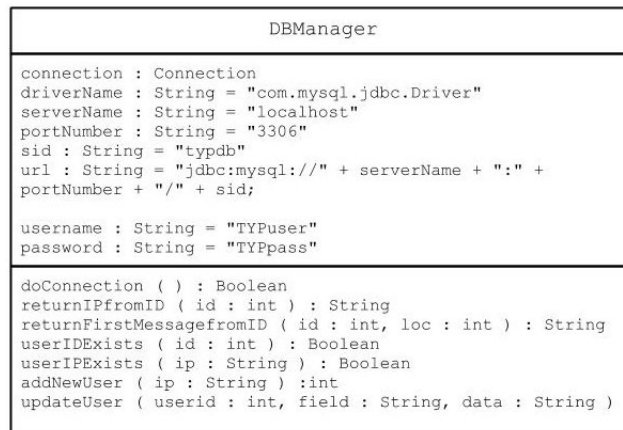
Figure 4.2: UML Class diagram for the DBManager class

4.3.4   Clients

The clients are how the users of the system interact and operate the system and act as a mirror image of the server. During the log in protocol, the server requires the users log in ID. This is mirrored for the client in the request for the user to input their user ID. The user wishes to send a message to another user, so enters the recipient ID and message. This is mirrored for the server by the receiving of the recipient ID and message, which the server saves at a specified location on disk corresponding to the added database entry. The client also allows the data sent to the server to be encrypted and can decrypt data sent to the client. These requirements of the client are better represented in the following list which outlines the capabilities of the designed client:

- Connects to the server via sockets

- Generates and stores its own keys

- Specify which other client the user wishes to send a data message to

- Retrieve the data message input from the user

- Complete the key exchange protocol required

- Encrypt the message to be sent

- Send the encrypted, user input, data messages to the server

- Receive encrypted messages from the server

- Decrypt and present to the user the received message

The client class contains the methods and protocols allowing it to complete the above tasks. The P.C client and Android application inherit these abilities from the client but are more specific, so are implemented according to their own rules and requirements. For example, to ensure that the operation of the application does not noticeably impact the performance of the smartphone, socket connections and data transfer will be performed as an asynchronous, background, task. This allows a P.C client to send a message, via the server, to another P.C client or an android application client. Different encryption (and decryption) methods can be implemented through a different instance of the P.C or android clients. For example, in Figure 4.3, Application 1 and P.C Client 1 implement a particular scheme and Application 2 and P.C Client 2 implement a different encryption scheme. So, P.C Client 1 has all the capabilities of P.C Client, with an added encryption method and so on for the other instances of P.C and Android Clients shown in the diagram.
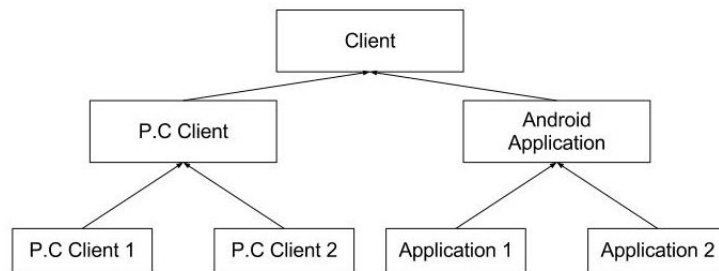

Figure 4.3: Basic inheritance hierarchy of the clients

Compatibility between the P.C and Android based clients is ensured through the use of the XML file structure. Figure 4.4 shows the design of the XML file that will contain all the data required for the message to be sent and received.

```
<?xml version="1.0" encoding="UTF-8"?>
<xmlmessage>
        <message id="1">
                <sender>User: 4</sender>
                <receiver>User: 8</receiver>
                <text>Hello</text>
                <other>null</other>
        </message>
</xmlmessage>
```

Figure 4.4: XML file layout.

The XML file creation and parsing (data retrieval) will be managed by separate management classes, instances of which will be used by each client. Figure 4.5 displays the UML class diagrams for these designed classes.
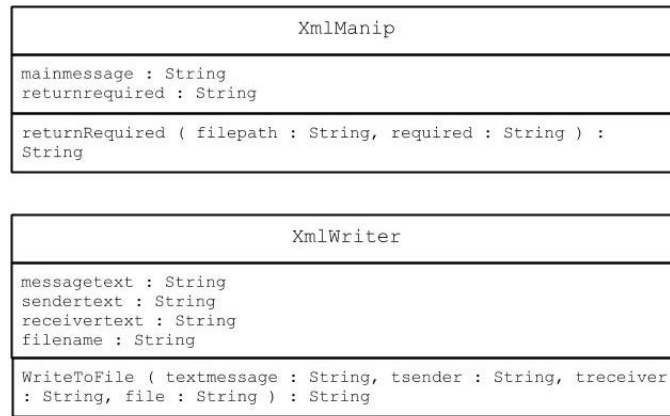
```
                           XmlManip
─────────────────────────────────────────────────────────
mainmessage : String
returnrequired : String
─────────────────────────────────────────────────────────
returnRequired ( filepath : String, required : String ) :
String
```

```
                           XmlWriter
─────────────────────────────────────────────────────────
messagetext : String
sendertext : String
receivertext : String
filename : String
─────────────────────────────────────────────────────────
WriteToFile ( textmessage : String, tsender : String, treceiver
: String, file : String ) : String
```

Figure 4.5: UML Class diagrams for the XML management classes

4.3.5 Key Exchange

The sharing of keys is required by all the encryption schemes described and used in this project. A key exchange protocol is required to ensure that the user a communication link is set up with is actually the intended user and not a malicious one. The protocol also ensures that the correct key is given to the appropriate user through the use of timestamps and identification names included with the key transfer. Figure 4.6 illustrates the key exchange protocol used, as described by William Stallings [22], which is presented bellow.
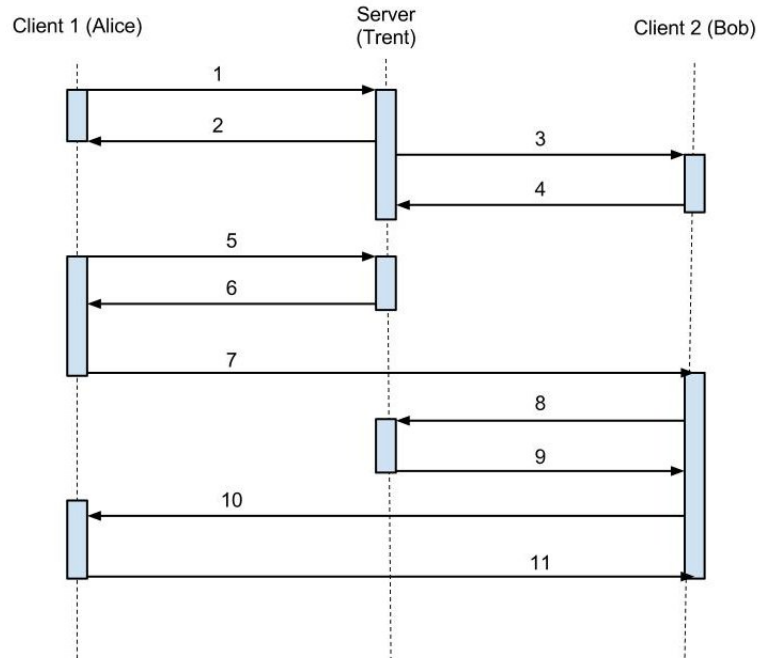

Figure 4.6: UML Sequence diagram showing the distribution of public keys

1. Alice registers her public key K_A with Trent

2. Trent sends his public key K_T to Alice

51

3. Bob registers his public key K_B with Trent

4. Trent sends his public key K_T to Bob

5. Alice sends to Trent: $Alice, Bob, Timestamp1$

6. Trent sends to Alice: $\{K_B, Bob, Timestamp1\}K_T^{-1}$

7. Alice checks Trent's signature on "K_B, Bob" and the timestamp, creates her nonce N_A at random and sends to Bob: $\{N_A, Alice\}K_B^{-1}$

8. Bob decrypts the message, checks Alice's ID and sends to Trent: $Bob, Alice, Timestamp2$

9. Trent sends to Bob: $\{K_A, Alice, Timestamp2\}K_T^{-1}$

10. Bob checks Trent's signature on "$K_A, Alice$" and the timestamp, creates his nonce N_B at random and sends it to Alice: $\{N_A, N_B, Bob\}K_A^{-1}$

11. Alice decrypts and sends to Bob $\{N_B\}K_B^{-1}$

(Steps 7, 10 and 11 passed through server and directed straight to client)

The notation

$$\{M\}K_x^{-1}$$

represents the encryption of message M with the public key of user x, which is decrypted using the private key of user x.

Once this protocol has been completed, each client has the other client's public key and the communication of encrypted data can commence. Steps 1 to 4 registers the public keys of the clients with the server, which are only required when a connection is made between two clients that are new to the system or have not yet made a connection to the server. Therefore, these steps will not be required for each new session initialisation. In practice, the protocol should be completed periodically to ensure the currency of the public keys of each user.

The protocol contains various techniques to ensure smooth and correct operation. For example, the use of timestamps gives a specific time frame in which the server should respond with the required key. If the delay is larger than a specified time then an error has occurred and the key should be sent again. The inclusion of a random number (nonce), for example, in step 10 is because the only way in which Bob could return Alice's nonce N_A is if Bob decrypted the message in which Alice included her nonce (Step 7). This message was encrypted using Bobs public key, so when Alice receives her nonce back, she knows that Bob must have sent it, as he is the only one that could have decrypted it.

Chapter 5

Software Development

5.1 Introduction

The design section showed how a full system could be laid out to achieve encrypted data communication between two clients. The development stage of this project is concerned with creating portions of this system in order to perform tests and analysis which would allow a full conclusion to be drawn. In this chapter, the software development process will be discussed and the final system presented, including the tools used and any development choices made.

5.2 Development Tools

5.2.1 Programming language

There were a multitude of programming languages that could be chosen to develop this project, each with their own benefits and drawbacks to certain tasks. For this project, there were two main programming languages to choose from concerning the development of the server and P.C clients, Java and C. C is a low-level, procedural based programming language. Constant, explicit management of pointers and memory allocations is required when developing in C. It also has no error handling capabilities. Java, on the other hand, is a high-level, object-orientated programming language. Pointers and memory allocations are effectively managed 'behind the scenes'. If an error occurs in Java then an exception is thrown which can be handled appropriately by the developer [29]. It can easily be seen that, for this project, Java is the best choice of language to use. This is enforced by the fact that Android applications are developed solely in Java, so developing the server and clients in Java is the logical choice.

5.2.2 Development Environment

Programming in Java can be completed using a text editor and the terminal. However, this requires the programmer to personally manage all aspects of the development, no matter how trivial or basic, and can lead to unnecessary errors. Development can be made easier through the use of a software development environment, specifically in this project, Eclipse [30]. The benefits of using Eclipse are as follows:

- Simple and visually appealing user interface

- Hierarchical project file and class management

- Code highlighting and completion

- Real-time error checking and reporting

- Integrated compiler

- Built in debugger

- Intelligent refactoring

- Library import management

- Plug-ins to allow android application development amongst others

As with developing software in Java, an integrated development environment can be used to aid in the development of MySQL databases called the MySQL Workbench. This environment allows the visual development, administration, design, creation and maintenance of MySQL databases using one piece of software as opposed to the more time consuming method of using the command line approach.

5.2.3 Smartphone

Using Eclipse to develop an android applications allows the use of an in-built Android emulator to run, use and test developed android applications directly from the environment. This, however, can be slow, tideous and prone to errors due to the fact that tasks such as networking have to be completed in an entirely different way. For this project, a physical smartphone was used as it was easily available and would improve the development process. Application debugging could take place directly on the smartphone through a USB connection to the development computer. The smartphone used was a Samsung Galaxy S3 [31] which, at the time of development, was a 'top of the range' smartphone running the Android operating system.

5.2.4 Version Control

As mention in the project progress report, various methods will be adopted to ensure that the project files and developed software are constantly and effectively backed up and that version control is maintained. The free services of Git/Github [32] and Dropbox [33] were set up on every computer and workstation involved in the development of this project. This allowed a constant back up of data from wherever development and project work was taking place. Furthermore, an external flash drive was dedicated to the task of providing a back up of data, in the event that an internet connection was not available.

5.3 Development Process

The software development process, also know as the development life-cycle, is a structure imposed on the development of a software product. Several models for how work should be undertaken and structured exists, each with various advantages and disadvantages. Although specified in the progress report, the chosen development model for this project will be reiterated. For this project the technique of test drive development in a plan-driven setting, as described by Ian Sommerville [34], will be employed.

The other option that was considered was an agile development approach. However, this method is best suited for a project with a dedicated customer so that constant communication can be maintained. Agile development encourages the constant evolution of a system to changing ideas of specifications and restricts the amount of documentation created. On the other hand, plan-driven development has a set plan for development which is followed closely to produce the end product and a firm set of documentation.

Test-driven development combines both testing and development. The software system is developed incrementally, along with a set of tests for that increment. The next increment is not started until the previous increment is fully tested and completed. In the context of this project, the implementation of the cryptographic schemes will not be started until the multi-client server is set up and can transfer data, for example. This ensures that a complete and error-free system is present to form the base of the next section to be implemented. Also, any important issues (design or programming related) are discovered and fixed early on in the development phase which, if not identified, could lead to disaster later on.

Another method that could be used is the waterfall method, where phases of the development cycle are completed in turn before the next stage is completed. This method, however, is very restrictive as movements back up the cycle are not allowed (like a waterfall). For example, testing is only started once the complete system is developed. If an error is found in the backbone of the system, fixing this issue could result in a complete

redesign on the system being required, wasting time and, in the development of a product to be sold, money.

5.4 Software Development

5.4.1 Overview

As described in full within the design section, the software system within this project has three main components:

Server - Facilitate the transfer of messages between two user clients of the system.

Database - Stores user information accessed by the user.

Clients - Sends and receives encrypted messages to other clients of the system. Is either P.C based or an Android application.

The result of the development of these components will now be presented. The full code listings of the project can be found on the enclosed CD and therefore actual programming code will only be present in this section if specifically required.

5.4.2 Server and Client

Due to the functions of the server and client that were developed for this project. it makes more sense to include the software description of these to aspects within the same section. The capabilities of this software pair are:

- Database connection by the server (see Database section)

- Connect via sockets

- Message encryption

- Message sending

- Complete Key exchange protocol

As both pieces of software were programmed using the java programming language, the methods in which some of the capabilities were implemented are similar, if not identical, so examples from only one of them will be given.

Connection   The network connection is established between the pair as soon as the client program is loaded. The client requests the connection and the server accepts which, if no errors occur, creates the communication link. The login protocol is then completed, as shown in Figure 5.1 and Figure 5.2.

```
Connection : true
Waiting...
Accepted connection : Socket[addr=/127.0.0.1,port=50465,localport=8500]
Is a user? y
YES
User ID is: 8
SELECT EXISTS (SELECT 1 FROM typdb.typusers WHERE id=8);
1
con.userIDExists result =true
IP Address is: 127.0.0.1
```

Figure 5.1: Login protocol from the server point of view

```
Tell the server that the user does have a user ID - 2 bytes
What is you user ID?
8
Tell the server the users user ID - 1 byte
Receive from the server whether the user ID is correct - 2 bytes
Successful log-in
```

Figure 5.2: Login protocol from the client point of view

Message Sending   Message sending via the sockets is completed using socket streams, which streams the input file through the socket and is received on the 'other end' of the socket and saved to disk. The sending and receiving of data files (conforming to the XML structure shown previously) is managed by a basic, separate class (*SendReceiveSocket.java*), making the main server and client classes clearer and more easily understandable, which aids in the development process.

Figures 5.3 and 5.4 show an example message being sent between two clients of the system.

If no new message is present for the user, a standard 'no new message' response is presented to the client.

Encryption and Decryption   The XML files, before being sent to the server (or client) are first encrypted. The methods for encryption and decryption are contained within a separate encryption/decryption class (*EncryptDecrypt.java*). When an instance of this class is created the cryptographic keys are initialised. either loaded from the storage disk or newly created if the client is a new user of the system (and no keys exist). The cryptographic object can then be passed the file to be encrypted or decrypted (depending on the method required) and the location of the required key, outputting the encrypted (or decrypted) file. Encryption and Decryption was completed in this project using

57

```
Do you wish to send a message? y/n
y
Tell the server that the user wants to send a message - 2 bytes
Enter receiver user ID
8
Tell the server that the user has the intended recipient registered - 2 bytes
Send to the server the user the following file is intended for - 1 bytes
Enter a message
hello there
File to be sent saved locally at: C:\Users/Tom/TestDoc/client8/toserver Mon Apr 22 20 04 27 BST 2013.xml
Encrypt the message
i is: 32
block: <?xml version="1.0" encoding="UT
i is: 32
block: F-8" standalone="no"?><xmlmessag
i is: 32
block: e><message id="1"><sender>8</sen
i is: 32
block: der><receiver>8</receiver><text>
i is: 32
block: hello there</text><other>1919191
i is: 32
block: 9191</other></message></xmlmessa
i is: 3
block: ge>1</other></message></xmlmessa
Encrypted message sent. 256 bytes
Receive notification from the server that message sending was successful - 2 bytes
Message sending successful
Do you wish to send a message? y/n
```

Figure 5.3: Client sending a message

```
Connecting...
Do you have a user ID? [y/n]
y
Tell the server that the user does have a user ID - 2 bytes
What is you user ID?
8
Tell the server the users user ID - 1 byte
Receive from the server whether the user ID is correct - 2 bytes
Successful log-in
Receive notification from server telling the user if there is a new message - 2 bytes
Encrypted message received from server
Saved at C:\Users/Tom/TestDoc/client8/receivedencryptedmessage.xml
Received 256 bytes
Acknowledgement sent. 11 bytes
Decryption of message:
block: <?xml version="1.0" encoding="UT
block: F-8" standalone="no"?><xmlmessag
block: e><message id="1"><sender>8</sen
block: der><receiver>8</receiver><text>
block: hello there</text><other>1919191
block: 9191</other></message></xmlmessa
block: ge>1</other></message></xmlmessa
Message is: hello there
From user: 8
```

Figure 5.4: Client receiving a message

the Java Cryptography Architecture (JCA). This architecture provides the implemented cryptographic methods with which encryption and decryption can be completed. The Java provided cryptographic methods were used because the actual implementation of the schemes is not the focus of this project. The focus of this project was to study the theory and mathematical concepts behind the cryptographic schemes and analyse the comparisons made between the techniques, particularly based on a smartphone. The way in which the schemes could be implemented would be a full project on its own. Imple-

menting two different schemes, without first thoroughly studying the various techniques to do so, could result in negative effects on the project results. For example, a certain implementation of generating random numbers for a certain scheme could significantly impact the running time of the scheme for reasons not known without thorough implementation research, where in actuality the running time is much quicker than all other schemes. Using the JCA allowed for a level playing field for all the schemes to be correctly compared to each other, without the worry of implementation comparisons.

Note: Included on the attached CD is an example of a file before and after encryption (see the *ReadMe.txt* file)

Key Exchange   The completion of the key exchange protocol is a vital part of the developed client/server system for this project. Looking back at the description of the protocol, various stages involve sending files between two users. This is done in the same way in which basic messages are sent, but is not made apparent to the user (is completed 'behind the scenes'). The protocol is split into three sections to be implemented;

- **Key exchange request (steps 1-7)** - User A (Alice) wishes to start communication with User B (Bob) so requests from the server Bobs key and sends to Bob a notification that key exchange is requested.

- **Receipt of request by Bob (steps 8-10)** - Bob receives the request,obtains Alice's key, adds Alice to the list of trusted users and sends to Alice his nonce (in continuation of the protocol).

- **Final Stage (step 11)** - Alice receives Bob's nonce and adds him to the list of trusted users.

The message file locations (messloc2 - messloc4) are used for the completion of this protocol. The file sent in stage 1 from Alice to Bob is saved in Bobs messloc2 location. When Bob logs in to the system, the server checks this location, notices that a key exchange request file is present and completes the next stage of the protocol for Bob. This is the same for any stage of the protocol for any user.

Figure 5.5 shows the first stage of the key exchange protocol implementation. It can be seen that key exchange is initiated if, when a user wishes to send a message to another user, no public key for the intended recipient is found by the users client. The filesize of files being sent across the network is output to allow analysis of the protocol later in the project.

```
Do you wish to send a message? y/n
y
Tell the server that the user wants to send a message - 2 bytes
Enter receiver user ID
8
You do not have user 8registered. Registration protocol will now be initiated
Tell the server that the user will start the registration protocol - 2 bytes
Alice creates the file: Alice, Bob, Timestamp
File sent to server. 17 bytes
Key received from server. 551 bytes
Acknowledgement sent. 11 bytes
Bob, Timestamp received (encrypted). 256 bytes
Acknowledgement sent. 11 bytes
File decrypted
block: 8 1366658933779
Check to see if the username returned is the same as the user requested (Bob)
Check to see if the timestamp returned is within an appropriate time frame
Notify the server that the username and timestamps are correct - 2 bytes
Tell the server which user the following file is intended for - 1 bytes
Create and save Alice's nonce - NonceA
Encrypt file to send to server - NonceA, Alice
i is: 10
block: 17407637 8
File sent. 256 bytes
Acknowledgement received. 11 bytes
Registration step 1 complete
Do you wish to send a message? y/n
```

Figure 5.5: Part 1 of the key exchange protocol

5.4.3 Database

Using the database design template, the database for the system was easily created through the MySQL workbench. Figure 5.6 shows how the creation of the table was completed. Constraints were also placed on various table columns in order to restrict the data that could be entered in an attempt to both protect the system from errors and to increase the access and data retrieval speeds of the system. For example, the IP address field can only contain 20 characters, a logical restriction as any valid IP address would not contain more than 20 characters.
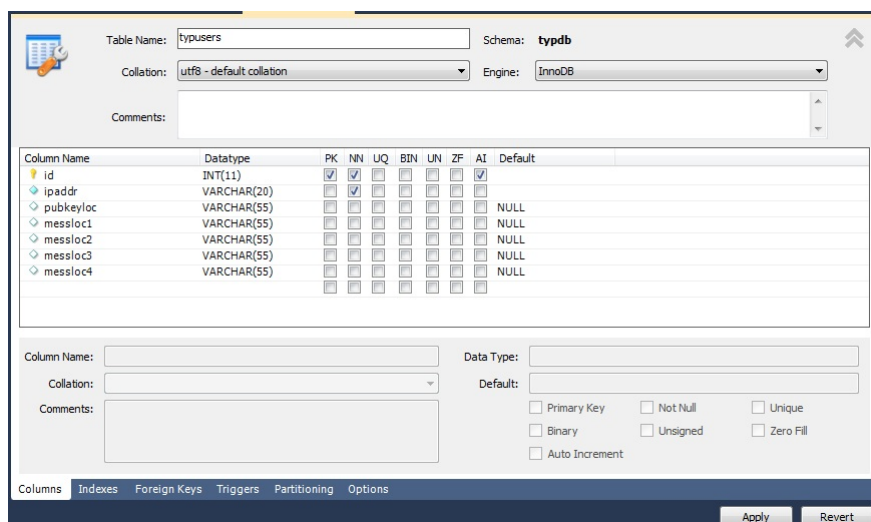

Figure 5.6: Database creation interface

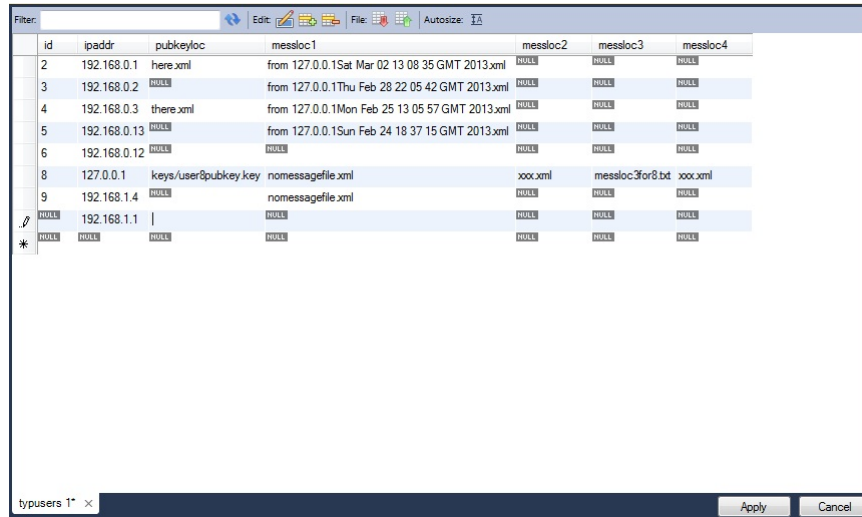The various abbreviated properties are:

60

PK: Primary Key - The primary key field of the database table.

NN: Not Null - This field of the database cannot be NULL so must contain an entered piece of data.

AI: Auto-Incremental - The value of this field is automatically entered and is an increment of the last automatically added value. This allows, in this case, a user ID to be generated by the system and not left to the user or administrator.

Note: 'messloc1' is for the message storage location, and 'messloc2-4' is for the storage locations of files relating to the key exchange protocol.

As the database is connected to by the Java server, some test data was required to be inserted into the database before the system as a whole was completed. The data was added into the system using the interface in the MySQL Workbench shown in Figure 5.7

Figure 5.7: Entry input interface

The java file *DBManager.java* contains all of the designed methods to allow the server to access the database. For example, the following code snippet shows the commands issued to create a new database management object, connect to the database and run a simple query to see if the given user ID exists in the database.

```
//create database management object
DBManager con = new DBManager();
//initiate connection to database
System.out.println("Connection:" + con.doConnection());
//use userIDExists method
boolean userCorrect = con.userIDExists(userID);
```

Data from the database is retrieved and updated using various MySQL query statements. For example:

SELECT EXISTS (SELECT 1 FROM typdb.typusers WHERE id=8);

selects all the database entries that exist in the table with an id equal to 8 and

UPDATE 'typdb'.'typusers' SET 'messloc4'='messloc4for8.txt' WHERE 'id'='8'

updates the field 'messloc4' for the entry with the id equal to 8.

These statements are generated by the database management class through the relevant classes specified in the design section.
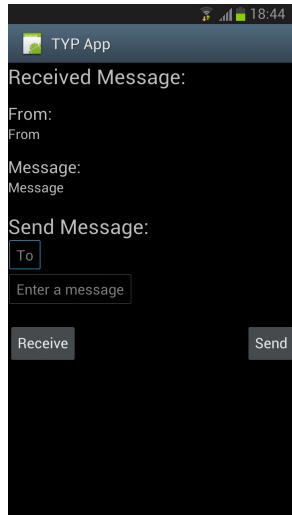
5.4.4 Application

The application development involved implementing various tasks and functions. Firstly, it had to connect to the server via sockets, which was completed using the same socket-based method as the P.C client. However, as explained in the design section, the network connection was established through an Async Task created within a connection handling class. The networking capabilities were implemented in the same way as the P.C client, but the use of the Async Task removed the workload from the main thread ensuring that the performance of the devices other features was not affected.

Again, message sending and receiving was completed in a similar way to that of the P.C client, however, as the application is based around a user interface and not the command line, the received data had to be processed in a different way. The elements that make up the user interface had to be updated with the received data message, retrieved from the xml file using the management class and passed to the corresponding screen. This can best be shown in Figure 5.8.

Message input is completed by entering the required data into the corresponding text input boxes on the application screen, as shown in Figure 5.9.

Once the user has entered the message they require, pressing on the 'Send' button sends the message and takes them to a screen that shows the message sent, displayed in Figure 5.10. This screen also shows the relevant data for the analysis of the implemented encryption schemes using the same method as the encryption within the P.C Server and Client. Timestamps placed before and after the various components of the cryptographic techniques enabled data to be produced that would allow a full analysis of implemented cryptography on smartphones to be completed. This will be discussed in full in the relevant analysis section.

(a) Before retrieve button pressed
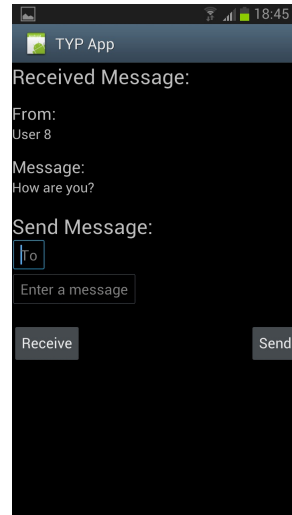
(b) After retrieve button pressed

Figure 5.8: Receiving a message in the Android application
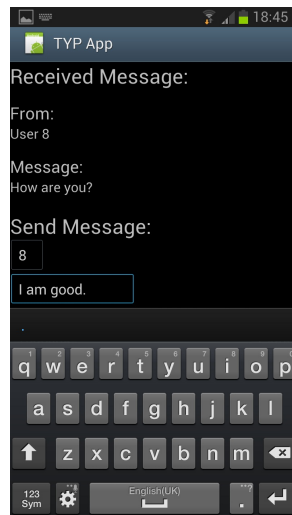
Figure 5.9: Inputting a message screen capture

5.5 Testing

As described previously, testing was completed alongside the development of the project in an incremental fashion. Testing is a very important stage in all software development projects as it brings to light any errors in the software and helps to ensure that the product meets its requirements and specifications. It also allows the software system to be validated and verified. Various techniques were used to test the developing system, which will be described here.
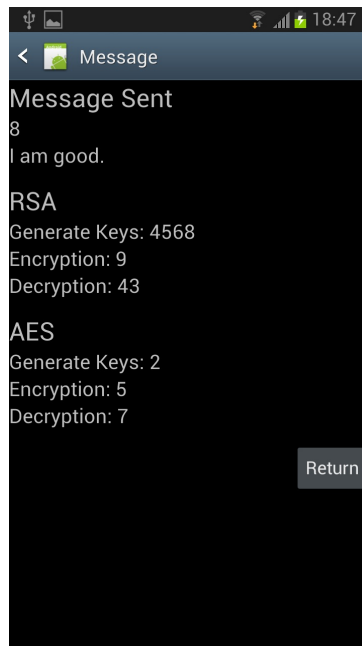
Figure 5.10: Final application screen capture

5.5.1 Testing tools

Within Android application development, there are two different types of errors that can occur. Firstly, the application may compile correctly but fails to carry out a particular function or operation and presents a system error message, shown in Figure 5.11. The application 'force closes' and is either restarted or shut down completely.

Figure 5.11: Application error message

The second form of errors are those where the application runs, but the results of a particular function are not as intended. An example in this project would be where a blank file is received from the server by the client, instead of the intended message.

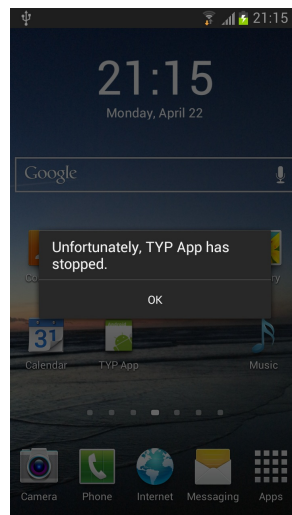The logging screen found within eclipse (LogCat) displays log messages sent from the application. These are both user and application defined messages. Data can be passed through from the applications and presented on screen to allow the application to be thoroughly tested. Figure 5.12 shows an example of the logging screen being used by the application (dalvikvm tag) or for testing (following two tags).

L...	Time	PID	TID	Application	Tag	Text
D	04-22 20:58:4...	1229	1233	com.thirdyearproje...	dalvikvm	GC_CONCURRENT freed 94K, 6% free 12503K/13191K, paused 4ms+3ms, tot 〇 al 35ms
I	04-22 20:58:4...	1229	1229	com.thirdyearproje...	from server path is:	/storage/sdcard0/Android/data/com.thirdyearproject.app/files/Downlo 〇 ad/fromserver Mon Apr 22 20 58 40 GMT+01 00 2013.xml
D	04-22 20:58:4...	1229	1229	com.thirdyearproje...	Error	Testing error message
D	04-22 20:58:4...	1229	1229	com.thirdyearproje...	libEGL	loaded /system/lib/egl/libEGL_mali.so
D	04-22 20:58:4...	1229	1229	com.thirdyearproje...	libEGL	loaded /system/lib/egl/libGLESv1_CM_mali.so
D	04-22 20:58:4...	1229	1229	com.thirdyearproje...	libEGL	loaded /system/lib/egl/libGLESv2_mali.so
D	04-22 20:58:4...	1229	1229	com.thirdyearproje...		Device driver API match

Figure 5.12: Eclipse logging screen example

Errors in network communication, socket establishment and other sections of the client or server systems are found through the use of java error Exception Handlers. Testing the other sections of the java based software products was completed using standard print methods, outputting important data locations, variable values and so on.

5.5.2 Testing Methods

The multitude of the testing of this system was completed in the form of integration testing, where the developed section of the system was thoroughly tested before the next section was developed, until the system was completed. The initial stages of testing involved unit testing, testing each individual component or method of a developed class, verifying that the section is correct according to its requirements. Unit testing is completed as soon as the particular unit has been encoded (after source code generation).

Validation testing was completed by analysing the output from the system and ensuring the data was recognisable and required by the user. As the system is created for the purpose of analysis and not with a user-base in mind, validation testing was not a large part of the overall testing phase.

System testing involved testing all of the various components of the system together to ensure the system operated correctly as a whole. This was the last form of testing that was completed. The main test that was carried out was that a new user of the system could be added to the database and successfully complete the key exchange protocol allowing the user to send and receive data messages to another user.

An example of specific tests that were carried out during the testing phases of this project were:

- The output of the encryption methods was identical to the input plaintext.

- The contents of a file sent across a socket connection was identical for both the receiver and sender of the file.

- The message output to the client (or server) matched the message that was sent.

- The expect data and file contents was sent and received in all stages of the key exchange protocol.

Chapter 6

Results

In order to fully and accurately present and conclude the results of this project, the questions that were presented at the start of this report, which form the basis of this project, will be studied individually. Each question will be answered using results from the research completed or the data gained from the developed system, allowing a final conclusion to be made.

6.1 Analysis

Do applications that facilitate the secure transfer of messages exist? Through research completed using the Google Play application market, two applications where found. Both applications facilitated the encryption and decryption of data (text) messages, however this was the only relevant feature of the application RSA Cipher Cat. On the other hand, the application Cloak SMS allowed both the encryption/decryption of messages and the sending/receiving of these messages. Cloak SMS was found to be the more popular application, through user reviews, ratings and number of users, with RSA Cipher Cat having a quite poor user size and review rating.

What are the cryptographic techniques behind these applications? The Cloak SMS application is based upon the AES symmetric-key algorithm whilst RSA Cipher Cat utilises the RSA public-key cyptographic scheme.

Are there other techniques which could also be used for this purpose? The chosen technique that could also be used for the purpose of encrypted data transmission is the asymmetric, Elliptic Curve based, cryptographic scheme ECC.

What is the performance of these styles applications in comparison to each other?

Key Exchange Every application of encrypted data communication requires the same initial operation; key exchange. Therefore, in order to fully analyse the system, a study into the costs of completing this protocol is required. Table 6.1 displays the data retrieved from the server and P.C client completing the protocol using the RSA encryption. The cost in bytes is the amount of data sent to and from the client and server (with respect to the client) for each part of the protocol. Example files would be those containing public keys, nonces or even simple acknowledgements. The figures shown in the development section (Figure 5.5) show how this data was collected. Within the encoded software, each time a file or piece of data was sent to (or received from) the server the size of the file was discovered and printed to the display, using the following commands:

```
//receive the file from the server
sendrecsock.ReceiveViaSocket(otherfp);

//initialise the file object
filesizefile = new File(otherfp);

//save the filesize within the appropriate variable
filesizesize = filesizefile.length();

//display the filesize for that file in bytes
System.out.println("Received protocol file"+otherfilepath+"-"
                    +Long.toString(filesizesize)+"bytes");
```

However, as each step of this protocol needs to be completed by every cryptographic implementation, the only possible way in which this result could be changed or improved is by the use of a different encryption scheme.

Implementation Comparison As shown in the research section, there where many possible factors that could be used to compare the two application styles and the cryptographic methods they used. A selection of these factors were chosen and used to create the following comparisons.

Firstly, a comparison was made between the key generation times of the two schemes. This was completed using timestamp outputs placed before and after the key generation

Section	Cost (bytes)
Part 1	1146
Part 2	1413
Part 3	563
Part 4	291
Total (Protocol)	3413
Sending a message	289
Receiving a message	293
Total (Send/Receive)	582

Table 6.1: Key Exchange protocol data usage

commands within the application. How the data was output can be seen in Figure 5.10 within the development section. Table 6.2 shows the results of this test.

(milliseconds)	**AES**	**RSA**
Generate Keys	2	781

Table 6.2: Key generation time results

A similar method was used to compare the encryption and decryption times of the two algorithms within the application, the results of which can be seen in Table 6.3

(milliseconds)	**AES**	**RSA**
Encryption	4	5
Decryption	9	45

Table 6.3: Encryption and Decryption duration times

It must be noted that the times are in milliseconds. Therefore, whilst the decryption using RSA takes 5 times as long as decryption by AES, this is still only 45 milliseconds, which would not impact the application operation enough to be noticeable to the user. The only result that would possibly be noticeable to the user would be the key generation time of the RSA algorithm. However, in practice this would be hidden behind a splash (loading) screen with other operations such as server connection initialisation or loading graphics.

Cryptographic Scheme Comparison   The main comparison that can be made between the cryptographic schemes themselves is the size of the key required for certain levels of security. This has an impact on the size of the data stored by an application, the encryption/decryption running times, the size of the encrypted file being sent and the battery and computational power consumed by the application. Table 6.4 shows the various key sizes of the three schemes at comparable levels of security (computational effort for cryptanalysis).

AES	RSA	ECC
56	512	112
128	3072	256
192	7680	384
256	15360	512

Table 6.4: Key size (bits) comparison table

This shows that 512 bit ECC key provides the same security as a 15360 bit RSA key and that for equal key lengths, the computational effort is similar ([22] page 344), reinforced in the paper by Yacine Rebahi *et al.* [35].

Can these applications or methods be improved upon?   As can be seen from the results and will be discussed later, there are ways in which the applications can be improved upon, if there was the demand for it. A combination of the AES and ECC schemes would provide the best form of security. ECC would be used within the key exchange protocol, to set up the communication link and to send and receive the AES key. The AES key would then be used to encrypt and decrypt the sent and received messages from then on. This is because, whilst AES provides the greatest security for the smallest key size, it is a symmetric key algorithm, so if someone were to gain access to the key, they would be able to read all messages encrypted by that key. Also, encrypting the whole message using a public-key method may be computationally infeasible for a large message (increased key size results in increased running times for larger messages).Therefore, a more secure way to exchange the key is required, which is why the ECC scheme is used. Utilising the ECC scheme would allow the safe and secure exchange of the AES key. The protocol is called the Elliptic curve Diffie–Hellman scheme, described by William Stallings [22].

More commonly, the RSA scheme is used in place of the ECC scheme, within the standard Diffie-Hellman protocol. RSA is found to be the more popular encryption method compared to ECC mainly because RSA is well established, having been present

in the cryptographic field for a longer time. RSA is also more easily understandable so would be the first choice for engineers that wish to quickly develop a secure system with no constraints on computational power or disk usage.

Can a conclusion be made about the current state of secure message communication and its future? The average smartphone has the capabilities to store upwards of 1GB of data and has sufficient processing power to complete multiple tasks without being noticeably slow to the user. The majority of smartphone users have contract plans that include between 250MB and 1GB+ of mobile data usage per month (data uploaded or downloaded through the mobile network). Because of this, the amount of data and power requirements of applications facilitating message encryption and transmission does not have any significant effect on the user in terms of cost (mobile data usage), storage use (saved data) and time. Therefore, it can be stated that the current state of secure message communication is appropriate and acceptable in terms of application usage impact for the user and the level of security provided by the schemes. However, the number of available applications is incredibly small, so there is definitely a space in the market for the development of more, publicly available, encrypted communication applications. This would also increase competition and thus the creativity between competing developers.

As the computing power and data usage allowance per user will only increase, the main factor that impacts the future of secure message communication is the cryptographic schemes themselves. As the RSA algorithm, for example, is more widely established and studied, the size of the key required to give sufficient security is becoming increasingly bigger. Therefore, as ECC provides smaller key sizes (and lower energy usage) than RSA, once the RSA algorithm implementation starts to impact the usability of a smartphone application, it should be replaced with the ECC based scheme.

6.2   Conclusion

In conclusion, it can be seen that, due to the smartphone power and data allowance available to the average user, a new or improved application that facilitates the secure transmission of data messages is not currently required. However, when the implementation of the current cryptographic schemes, particularly RSA, becomes such that the usability of the applications is effected, the alternative approach of using ECC to encrypt and transfer an AES key is advised.

This result is an original contribution to the field of mobile application development as well as computer science, as research found that this conclusion has not been recently reached in this way before. As smartphone capabilities and design is rapidly changing,

currency is very important. The results from this project could be used to help lead or focus the development of encryption related applications, as well as form a basis for further work. This is the reason that the outcome and achievements of this project feature heavily within the Positives section of the Evaluation chapter.

Chapter 7

Further Work

Due to the nature of the project, a non-negotiable deadline forced a fixed time frame for the project, resulting in a few ideas or areas that could be developed or continued further. The project could also be taken in many different directions due to the depth and breadth of the field the project falls into. The further work that could be completed concerning this project will now be discussed.

7.1   Public-Ready Product

As the resulting software of this project was developed to study and analyse the 'workings' of an encrypted data communication application, the system being appropriate for a large public user-base was not focused upon. In order for the system to cope with a larger amount of users, various changes and modifications would need to be made, some of which have already been discussed. The main changes that would need to be made would be:

- Added message storage database table to store message locations associated to every user.

- Improved user identification such as unique email address entries for each user.

- Higher user security through the use of password protected user accounts.

- Host the server and database on an external, paid for, server that can cope with higher demands, data security and file storage.

In terms of making the system suitable for public users, the marketability and aesthetics of the client applications should be considered. A brand would need to be created in order to market the client applications and increase product recognition and publicity.

Thought would also have to be given to the monetary value of the product, including profit projections for example, and any further development costs required. This would however fall under more of a business development project, so research into this field would firstly be required before the further work could be started. Designing and developing an industry worthy user interface would also be a major requirement if the further work into producing a profitable product were to be completed [36]. A good user interface allows the user to intuitively identify the task they want to complete and how they go about completing it, encouraging them to reuse the application and recommend it to others. An example of the factors that would need to be considered when developing an appropriate user interface would be:

- Elements positioned in the location the user expects

- Colour coded buttons or sections used well

- Help and suggestion tips

- 'Next step' indicators

- Smooth flowing page transitions and loading times.

7.2 System Uses

The uses of the system could also be expanded upon and continued, as the operation of the system is not limited to just secure message sending. Possible continuations into different uses of the system could be:

Bank communication – The system could be set up to allow the communication of information to and from user bank accounts, enabling the completion of bank account related tasks such as money transfers. However, as products that perform this task exist already, a way in which the result could be original would have to be considered. Thorough and rigorous product testing and a study of the law would be required, however, as bank account management is a very sensitive area of computing as a result of fraud, theft and numerous security attacks.

Secret sharing - Secret sharing or zero-knowledge proofs could be another possible application of the system. The basis of secret sharing is that a number of users each have a piece of information that, when put together with the pieces from the other users, forms some final result which cannot be reached with any subset of the total pieces of information. An example of this would be a treasure map, where the location of

the treasure can only be found when all the pieces of the map are put together [37]. Zero-knowledge proofs are where a number of users wish to find a particular result or comparison between them, but do now wish the other users to know what information they have. An example of this would be a number of company owners that wish to find out who has the largest salary, without the owners finding out the salaries of each other. Applications of these techniques regarding this project could be friends wishing to compare exam grades to find who achieved the best results (zero-knowledge proof) or the share-holders of a company selling all of their shares only when every member agrees to (secret sharing). This could be completed using the same smartphone based data communication system but with a different, specific, purpose.

7.3   Study of Implementation

A thorough investigation into the actual implementation of the various cryptographic schemes could provide a possible area to continue within this project. This would involve implementing a scheme, studying various computational comparison factors of the implementation and developing an improvement to the implementation. Whilst this would be a possibility for further work, the amount of original contributions that could be made as a result of this work is insignificant, due to the already well developed and studied field of algorithm implementation and improvement.

Further work of this type could also be accompanied with an in depth study of the related cryptanalysis.

Chapter 8

Evaluation

This chapter acts as both an evaluation of the project and as the authors self-assessment, discussing the positive and negative aspects of the project, as well as any lessons learnt.

8.1   Positives

The main positive results of this project are the overall achievements made. Each objective was successfully completed allowing a system to be developed and utilised in the analysis of very current and relevant applications. The fact that a solid conclusion, that is both useful and relevant to mobile application development and the field of computer science as a whole, was reached is a great positive of this project. Being able to state a conclusion based upon a developed system and valid set of results was the ultimate goal of this project. The project can therefore be deemed a success.

Although covered later in this chapter, it is worth noting that the skills and knowledge learnt was a large part of the positive results of this project. This included the study of intricate and challenging mathematical ideas as well as the learning of new programming methods and techniques.

8.2   Negatives

Firstly, the size of the cryptographic field was such that the amount of material was overwhelming, resulting in a focus for the project being difficult to find and thus a slow start to the project. Throughout the project many different schemes and algorithms were found to be associated with so much other material concerning the scheme being studied that a whole project could be centred around that method alone.

Secondly, due to time constraints, the duration of other project areas and other module commitments, the cryptographic techniques used were unfortunately not personally

implemented but instead implemented using java provided libraries. Whilst this is a negative of the project, implementation of cryptographic schemes was not the focus of this project. The understanding of the techniques and the conclusions gained was in fact the focus of this project, which were both achieved successfully.

As can be seen through the amount of research that was required, the learning of new material and techniques posed a challenge within this project. Cryptography had previously not been covered, either in lectured material or personal study, so had to be learnt 'from the ground up'. Programming techniques such as database creation and java socket programming also had to be learnt from scratch.

Lastly, personal time management created a number of problems and negative effects throughout the project. This was centred around the fact that the assignment of task completion lengths was poor and not always adhered to. As with the learning of new material, even though personal time management is described as a negative, it is also features largely in the lessons learnt section.

8.3   Lessons Learnt

As mentioned previously, both time management and the learning of new material were negatives of this project. However, they are also valuable lessons that have been learnt (and thus are positives also). Having the opportunity to plan and timetable of a full project from start to finish has highlighted the issues present with personal time management which, now that they have been brought to light, can be improved and work upon. Also, the techniques for learning a new topic or programming concept without the assistance of a lecturer could be put into practice and improved. This led to a greatly increased knowledge of mathematics and computer science. These are valuable lessons learnt, which far out weigh any negative impacts.

Developing a full system that also required a full set of documentation resulted in the learning of much better coding and software development practices. For example, the management of code into classes and system folders was improved, as well as debugging and incremental development.

8.4   Conclusion

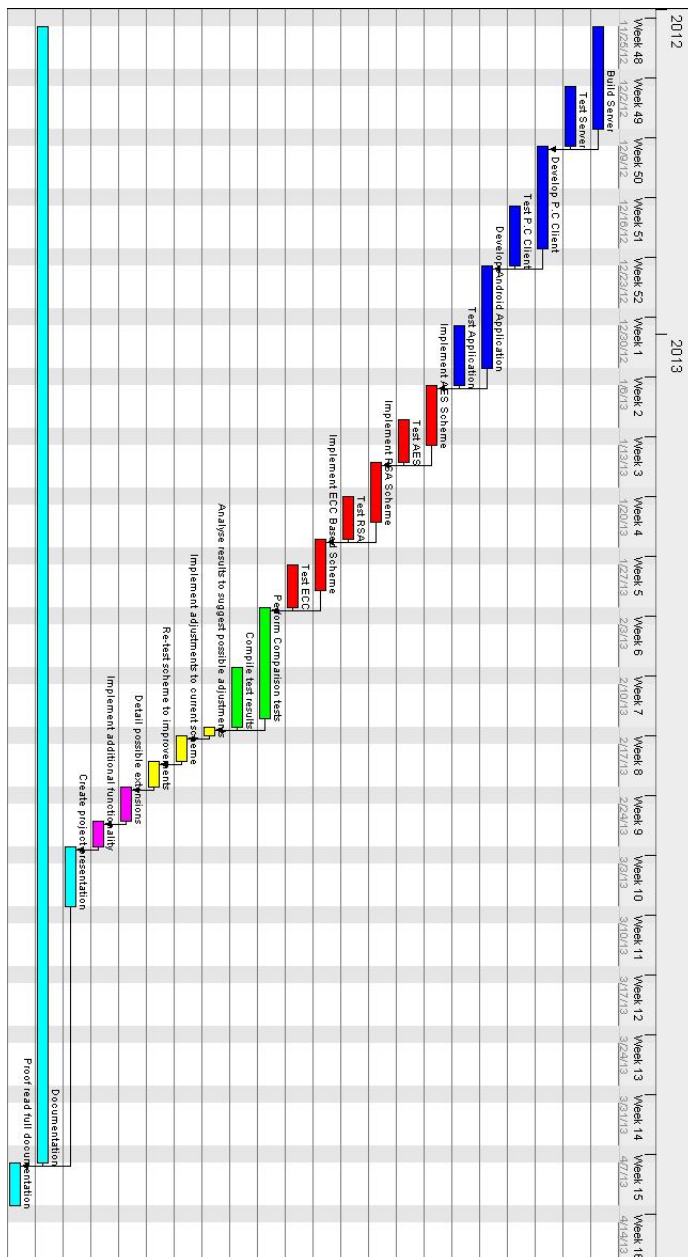In conclusion, this project fully achieved its goals and is a success.

Chapter 9

Acknowledgements

I would like to thank my project supervisor Marcin Jurdzinski, whose help and guidance throughout this project has been greatly appreciated.

Appendix A

Gantt Chart

Bibliography

[1] Allen, S [et al]. Pro Smartphone Cross-Platform Development: iPhone, Blackberry, Windows Mobile and Android Development and Distribution. Apress; 2010.

[2] Apple Inc . Apple iPhone iOS;. [Online](URL `http://www.apple.com/uk/ios/`). (Accessed April 7 2013).

[3] Microsoft Coporation. Windows Phone 8;. [Online](URL `http://www.windowsphone.com/en-gb`). (Accessed April 7 2013).

[4] Google Inc . Android;. [Online](URL `http://www.android.com/`). (Accessed April 7 2013).

[5] Google Inc . Google Play Store;. [Online](URL `https://play.google.com/store`). (Accessed April 5 2013).

[6] Mollin RA. An Introduction to Cryptography. 2nd ed. Chapman & Hall; 2007.

[7] Data Protection Act 1998. London: Stationery Office; 1998.

[8] Rowland, D [et al]. Information Technology Law. Taylor and Francis; 2011.

[9] Kirk J. UK Data Encryption Disclosure Law Takes Effect; 2007. [Online](URL `http://www.pcworld.com/article/137881/article.html`). (Accessed April 10 2013).

[10] Liang D. Introduction to Java Programming. Sixth ed. Boston: Pearson; 2007.

[11] Graba, J. An Introduction to Network Programming with Java. 2nd ed. Springer; 2006.

[12] Google Inc . Android Developers;. [Online](URL `http://developer.android.com`). (Accessed April 7 2013).

[13] Gargenta M. Learning Android. O'Reilly; 2011.

[14] Pozrikidis C. XML in Scientific Computing. Chapman and Hall; 2012.

[15] Refsnes Data. w3schools.com - XML Tree;. [Online](URL `http://www.w3schools.com/xml/xml_tree.asp`). (Accessed April 9 2013).

[16] Oracle. MySQL;. [Online](URL `www.mysql.com`). (Accessed April 9 2013).

[17] Parsian, M. JDBC Metadata, MySQL, and Oracle Recipes: A Problem-Solution Approach. Apress; 2006.

[18] Gemalto. Are SMS text messages safer than emails?;. [Online](URL `http://www.justaskgemalto.com/us/communicating/tips/are-sms-text-messages-safer-emails`). (Accessed April 11 2013).

[19] Medlin H. Cloak SMS; 2011. [Online](URL `https://play.google.com/store/apps/details?id=sms.encryptor.v2.free&hl=en`). (Accessed April 5 2013).

[20] Miasoft. RSA Cipher Cat; 2011. [Online](URL `https://play.google.com/store/apps/details?id=jp.co.Miasoft.RSACipher&hl=en`). (Accessed April 5 2013).

[21] Mao W. Modern Cryptography: Theory and Practice. Prentice Hall; 2004.

[22] Stallings W. Cryptography and Network Security: Principles and Practice. 5th ed. Pearson; 2011.

[23] VHDL. Advanced Encryption Scheme Student Wiki;. [Online](URL `http://vhdl.pbworks.com/w/page/4302806/Progetto%20AA%2008-09%3A%20Advanced%20Encryption%20Standard`). (Accessed April 13 2013).

[24] Flylib com. The AES cipher;. [Online](URL `http://flylib.com/books/en/3.190.1.55/1/#!/exjun_`). (Accessed April 13 2013).

[25] Rivest A R ; Shamir, Adleman L. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM. 1978;.

[26] Schneiers B. Applied Cryptography: Protocols, algorithms and source code in C. 2nd ed. Wiley; 1996.

[27] Jobanputra, N [et al]. Emerging Security Technologies for Mobile User Accesses. eJETAorg. 2009;2(4). Available from: `http://www.ejeta.org/issue-v2-n4/ejeta-v2-n4-2.pdf`.

[28] Bui MD. Real-time object uniform design methodology with UML. Springer; 2007.

[29] Dev R. 10 major differences between C and Java;. [Online](URL `http://www.durofy.com/programming/10-major-differences-between-c-and-java/`). (Accessed April 15 2013).

[30] Eclipse Foundation. Eclipse;. [Online](URL `http://www.eclipse.org/`). (Accessed April 16 2013).

[31] Samsung Electronics. Galaxy S3 Specifications;. [Online](URL `http://www.samsung.com/global/galaxys3/specifications.html`). (Accessed April 13 2013).

[32] GitHub, Inc . GitHub - Social Coding;. [Online](URL `https://github.com`). (Accessed April 10 2013).

[33] Dropbox, Inc . Dropbox;. [Online](URL `https://www.dropbox.com`). (Accessed April 10 2013).

[34] Sommerville I. Software Engineering. Ninth ed. Pearson; 2011.

[35] Rebahi, Y [et al]. Performance Analysis of Identity Management in the Session Initiation Protocol. ACS International Conference on Computer Systems and Applications. 2008;.

[36] Johnson J. Designing with the Mind in Mind. Elsevier; 2010.

[37] Vocking, B [et al]. Algorithms Unplugged. Springer; 2011.