ROYAL HOLLOWAY, UNIVERSITY OF LONDON

Enigma: Prime Numbers and Cryptosystems

by

Adam Mulligan

A final year project submitted in partial fulfillment for the degree of Bachelor of Science, Computer Science

in the
Department of Computer Science

January 2012

"Anyone can design a security system that he cannot break. So when someone announces, Heres my security system, and I cant break it, your first reaction should be, Who are you? If hes someone who has broken dozens of similar systems, his system is worth looking at. If hes never broken anything, the chance is zero that it will be any good."

Bruce Schneier, *The Ethics of Vulnerability Research*

Abstract

Modern cryptography allows us to perform many types of information exchange over insecure channels. One of these tasks is to agree on a secret key over a channel where messages can be overheard. This is achieved by Diffe-Hellman protocol. Other tasks include public key and digital signature schemes; RSA key exchange can be used for them. These protocols are of great importance for bank networks.

Most such algorithms are based upon number theory, namely, the intractability of certain problems involving prime numbers. The project involves implementing basic routines for dealing with prime numbers and then building cryptographic applications using them.

Acknowledgements

With thanks to my project advisor Yuri Kalnishkan, and the RHUL Department of Computer Science for three years worth of knowledge and experience.

Contents

List of Figures

List of Tables

Abbreviations

AES **A**dvanced **E**ncryption **S**tandard

Chapter 1

Introduction

1.1 What it's about

Secrecy has always been of great importance, not just in modern society, but throughout history. Until very recently, Cryptography was consigned to the sending and receiving of messages, generally using pen and paper. However, increasingly cryptography – the study and implementation of techniques for secure communication – is becoming more vital to the smooth running of even basic systems, whether it's the cliché example of military secrets or simply a micro-payment for an online service. Initially, the usage of provably secure and efficient cryptographic algorithms was limited to governments and related contractors, however the advent of encryption standards, and more relevantly, the creation of public-key cryptography mechanisms, has pushed it in to a wider field of use as researchers gained a better understanding of the area.

1.2 Goals and Intentions

The primary goal of this project can be found in the title and abstract: to research, discuss and create algorithms within or related to the field of prime numbers. To complete this task I will be developing my thoughts and discoveries regarding prime numbers as this report progresses, as well as producing a number of deliverables in the form of software applications, test results and statistics.The topic of number theory is in itself fascinating, and extraordinarily vast, however I will only be scratching the surface.

Nonetheless, I hope to explain throughout this project how prime numbers have become such an important part of modern cryptography, and what they can be used for.

1.3 Project Summary

I will be splitting the project in to four main parts, excluding this report:

1. **Prime numbers in software**
   A discussion of how prime numbers can be efficiently produced programmatically, and software to prove it.
2. **Algorithms**
   The development of algorithms using prime numbers, such as RSA, and complementary cryptographic algorithms such as AES. Alongside this, the development of non-major algorithms in the field of prime number based cryptography that still present an academically interesting concept.
3. **Final Application**
   The production of a software program that utilises the implemented cryptographic algorithms in section 2 to display their efficacy in a real world application.
4. **Cryptanalysis**
   Taking the algorithms created and comparing them with official implementations for statistical analysis, alongside researching and performing "attacks" on them to prove or disprove the implementation's cryptographic security.

The primary algorithms to develop are:

- Asymmetric
  - RSA
  - Diffie-Hellman
- Symmetric
  - AES

- Identification and Authentication
  - RSA Signing
  - Digital Certificates

Other algorithms will be discussed and produced alongside these, however they will not be used in the final application testbed and are purely for research interest. These can be found listed in the contents in the relevant sections.

1.4 Other Information

This document was written and composed with LaTeXusing *Taco Software's Latexian*. The LaTeXsource code for this report should have come bundled with the project documentation and files, however if you are unable to retrieve it please see section 1.4.1.

Eclipse Indigo[1] was used for Java development, the primary language used in this project, along with *MacVim*[2] for general source and text editing.

Git with *GitHub*[3] was used for source control, with GitHub Issues used for bug tracking. *Trello*[4] was used for idea and to-do management.

1.4.1 A note on openness

As with any field of study, the quality of research and development is dependent on the open distribution and sharing of ideas. This is *particularly* important with regards to cryptography. As said, cryptography was once reserved to government and research was conducted in secrecy. The open sharing of relevant information in this field is not just for the furthering of knowledge, but also to allow others to inspect and examine algorithms, a process that drastically improves the security of a system. As such, the entirety of this project is licensed under the **GNU General Public License version 3 or greater** and is available for access publicly online.

[1] http://eclipse.org
[2] http://code.google.com/p/macvim/
[3] http://github.com
[4] http://trello.com

1.4.2 Project Repository

If any part of this report is missing, you believe that you do not have a full access to the source code discussed in this document, or if any files have been lost, it is available for full download at `http://cyanoryx.com/files/project.zip`.

Chapter 2

Cryptographic Primitives

2.1 Basics of Information Security

Despite how it is portrayed or colloquially used, Information Security is an entirely different concept and area of study compared to Cryptography. It might seem simple enough to implement a basic cryptographic protocol involving encryption and decryption, however to introduce this into a system and expect the information to be secure, is foolhardy. Cryptography is a *means* to providing information security when following certain rules and guidelines not the be all, end all solution. An understanding of information security, and the related issues, is necessary.

This can be proven using historical evidence: throughout history many complex systems of mechanisms, rules, and protocols have been developed to introduce information security to a system. As with modern day security, this cannot be achieved entirely through mathematical and cryptographic means – it is more than just computational intractability.

As such, stringent criteria for developing secure systems and protocol have been introduced. While institutes such as *The British Computing Society* and *Association for Computing Machinery* ensure their members follow a professional code of ethics, just as a doctor might, these information security criteria are of separate and equal importance. Indeed, there are now several international organisations that exist solely for the overseeing of cryptographic research and development (See: *International Association for Cryptologic Research*).

Often, as we will see, cryptographic systems are simplified for the purposes of presentation particularly for textbooks. This will be discussed further later, with regards to the differences and difficulties involved in developing systems that do not just follow a mathematical "recipe," but also include information security values and other subtleties.

The overall method of dealing with, and ensuring, information security is known as risk management. This encapsulates a large number of countermeasures (including cryptography) that reduce the risk of vulnerabilities in, and threats to, systems. We will only be encountering and discussing the technological areas of mitigation, however some of the solutions include[1]: access control, security policy, physical security, and asset management.

2.2 Objectives

As said, secure systems should follow a guideline, or set of criteria, that ensure the security and integrity of data stored and input. A clear and concise specification should be developed, that will aid the designer in selecting the correct cryptographic primitives, but also help the engineer implement the protocol correctly. There are many of these criteria, however each is derived from four primary objectives:

1. **Confidentiality** is the ability to ensure data is only accessed by those who are allowed to. Maintaining confidentiality of data is an obligation to protect someone else's secret information if you have been entrusted with it.

2. **Authentication** involves identifying both entities and data. Two or more entities wishing to communicate or transmit information to one another must identify each participant to ensure they are who they claim to be - this is known as entity authentication. Data received must be authenticated to ensure the validity of the origin, date sent, contents, etc - this is known as data origin authentication.

3. **Non-repudiation** prevents an entity from denying previous actions they have committed.

[1]For an excellent resource regarding information security, both technical and non-technical, see *Security Engineering*, Ross Anderson

4. **Data Integrity** is how faithfully data compares to it's true state, i.e. proving that a data object has not been altered.

2.3 Key Concepts

As with any discipline, there are a number of fundamental concepts that need to be thoroughly defined. This section will cover the definitions of basic information security and cryptography related concepts. While Computer Scientists and Mathematicians, unlike Biologists, tend to abstract ideas using existing words such as *normal* which ultimately cause confusion to those trying to understand the area of study, the field of Information Security fortunately uses phrases that are succinct and aptly describe notions.

TODO: er, do this bit.

2.4 Primitives

As we discussed in the Objectives, there are certain criteria that must be met for an application to be considered as secure under Information Security guidelines. Excluding physical and psychological measures, there are a number of methods to be implemented cryptographically to guarantee security.

2.4.1 Encryption

Being what is seen as the very 'core' of cryptography, we have defined encryption many times already and what the term means in terms of a process should be apparent. However encryption takes many forms, with each having an appropriate situation for it to be used.

2.4.1.1 Symmetric Key Encryption

Primarily, we will use symmetric key encryption algorithms to encipher data that is to be transmitted between entities.

Mathematically we can formally define symmetric encryption as:

For a message M, algorithm A and key K,

$M' = A(K, M)$

and thus:

$M = A'(K', A(K, M))$

where $A' = A$, $K' = K$ in a symmetric algorithm.

2.4.2 Key Agreement

Key agreement, or key exchange, primarily ensures data integrity and confidentiality. By preventing an attacker from discovering encryption keys used on transmitted data, the attacker should be unable to feasibly read confidential information or modify it (the latter is not entirely true, it may be possible in some cases to modify encrypted data, however we will discuss that later in *Symmetric Cryptography*).

While it might be easier to use asymmetric techniques to encrypt data for transmission, thus allowing us to distribute keys as cleartext, it is slow and inefficient for large quantities of data, such as in an instant messaging application. Because of this, it is prudent to implement an efficient symmetric key encryption algorithm, and share the key (known as a session key) with other entities. However, it would be trivial for an attacker to launch a man-in-the-middle attack and gain access to the encryption keys during the initiation of the conversation, allowing the easy and undetectable decryption of all transmitted messages. As such, session keys will need to be distributed using a key exchange protocol.

2.4.2.1 Key Distribution Centre

The simplest solution is known as a Key Distribution Centre (KDC), which involves the use of a trusted third party (TTP). It is easiest to explain using an example. Alice and Bob are users of a system, attempting to securely communicate. Each share a key securely with third-party Trent (somewhat amusingly, this algorithm does not include

how these keys should be shared. We can assume that it was perhaps conducted through an in-person meeting of entities, or other means), who stores each key.

1. Alice initiates a conversation with Bob.
2. Alice requests a session key from Trent, who makes two copies of an identical key and encrypts one with Alice's stored key, and another with Bob's.
3. Alice receives both encrypted keys, and sends the appropriate one to Bob.
4. Alice and Bob decrypt their session keys, leaving both with a shared key.
5. Alice and Bob can now encrypt and decrypt data using the same key.

2.4.2.2 Asymmetric Cryptography

As can be obviously seen, using a KDC is dependent entirely on the ability of two entities to have previously, and securely, shared an encryption key with a TTP that is known to both entities. A solution to this is to eliminate the third party, and use an asymmetric algorithm to share keys directly. As defined in the *Key Concepts* section, asymmetric cryptography allows Alice to share a public-key, with which any entity can encrypt data that can only be decrypted using Alice's private key.

1. Alice and Bob share their public keys using a readily available database.
2. Alice downloads Bob's key, and vice versa.
3. Alice generates a session key, encrypts it using Bob's public key and sends it to Bob.
4. Bob can now decrypt the session key using his private key, resulting in both parties being in possession of a secure session key.

There is a security risk: how can you verify that the entity that sent the encrypted key is indeed the one with which you are trying to communicate? It would be easy for an attacker to encrypt their own session key with Alice's public key, and claim that the key is from Bob. The attacker would then be able to decrypt any messages intended for

Bob. The solution to this issue is the use of digital certificates, and signatures, which will be discussed in *Authentication*.

We will discuss specific algorithms further on, however it is worth pointing out that the current public-key algorithms used for these purposes are *RSA* (Rivest, Shamir, Adleman) and the*Diffie-Hellman Key Exchange* (not defined as a public-key algorithm, however it uses the sharing of public cleartext values). There exist a number of interesting algorithms that implement the public key architecture which will also be considered later.

There are other considerations in key management to ensure confidentiality and integrity. Some provisos exist such as changing the key for each session to ensure perfect forward secrecy, however these are trivial to implement and can be considered as part of the overall application security development.

2.4.3 Authentication

There are two types of authentication that are required in a secure application

1. Message authentication
2. Entity authentication

both of which require different protocols and algorithms. These terms have been defined in *Objectives*.

The methods of entity authentication are an interesting topic in themselves, with many, many protocols having been researched and created as the community tries to find a method that is both secure and easy for an entity to use. Some examples are: passwords, two-factor authentication, PINs, smart cards, biometrics, and so on. These are outside of the scope of this project – we will be implementing two broad, yet specific methods of authentication.

2.4.4 Digital Signatures

Digital signatures, as we will see, encompass three of the information security criteria: authentication, data integrity and non-repudiation (a sender cannot claim they did not

send the message). A digital signature is a string that connects a message with its originating entity.

When transmitting a message, the sender signs the message with their private key which can then be verified with their public key, which should be available to the receiver.

1. Alice signs her message A with her private key.
2. Alice sends message A with signature S to Bob $(A|S)$
3. Bob retrieves Alice's public key from an available database, and verifies signature S

The first and most common implementation of digital signatures is RSA.

2.4.5 Public-key Certificates

Digital signatures and a public-key infrastructure, however, are not enough by themselves. A very simple attack can be orchestrated similar to that during key agreement: Mallory, the attacker, could sign a modified message A with her own private-key and then distribute her public-key, claiming it to be Alice's. To counter this, we can introduce a trusted third-party, known as a Certificate Authority (CA), who signs Alice's public-key with their own private-key. Most trusted CA's public keys come bundled with software such as browsers and operating systems.

1. Certificate Authority signs Alice's public key with their private key.
2. The CA distributes its public key with major software.
3. Bob receives Alice's message and signature, as well as her public key and signature.
4. Bob verifies Alice's public key with the CA's public key, and then verifies the message.

2.4.6 Hashing

While not of direct relevance in information security, hashing plays a significant part in cryptographic systems and thus is included as a concept to be implemented. Formally, we can define a hash function as mapping a large domain to a smaller range - in the case of data, mapping a set of bytes to a unique identifier with a set length. A hash function, at the very basics, takes as input a message and produces a fingerprint, or digest, of the input. Within the field of cryptography, they are used for message authentication and data integrity.

This is to say, given a domain D and range R for $f : D \rightarrow R, then |D| > |R|$. This is a many-to-one relationship, the downside of which means that collisions can occur – two input strings resulting with the same output string – however, this varies between algorithms, the more modern of which are less likely to result in collisions.

A hash function can be classed into two categories: keyed and unkeyed, taking both a message and secret key and taking just a message, respectively. Two conditions are necessary for a hash function to be effective:

1. compression – the function f maps input a of arbitrary length to an output of fixed length, n.
2. complexity – it must be easy to compute $f(a)$

Most commonly used are unkeyed, one-way hash functions. Some examples of which are: SHA-1, and MD5. In cryptography, hashes are commonly used for data integrity in combination with digital signatures. A message is hashed, and the fingerprint produced is signed by the entity. There are some algorithms designed specifically for this purpose, known generally as Message Authentication Codes (MACs) and Manipulation Detection Codes (MDCs).

A sample of each of these four primitives will be implemented further in the report.

2.5 Mathematics

This section will cover some of the basic mathematical concepts that will be used throughout the report, and form a foundation of understanding for the more complex abstract methods that will be used.

2.5.1 Notation

1. $\mathbb{Z}$ is the set of all integers.
2. $\mathbb{R}$ is the set of all real numbers.
3. $[a, b]$ is the set of integers n such that $a \leq n \leq b$.
4. $|A|$ is the cardinality of a finite set, i.e. the number of elements.
5. $n \in A$ denotes that an element n exists in set A.
6. $A \subseteq B$ denotes that set A is a subset of set B.
7. $A \subset B$ denotes that A is a subset of B, but $A \neq B$.
8. $\lceil a \rceil$ is the smallest integer greater than or equal to a.
9. $\lfloor a \rfloor$ is the largest integer less than or equal to a.
10. A function (mapping) denoted as $f : A \to B$ signifies that every element a in A is mapped to exactly one element b in B. This can also be written as $f(a) = b$.

2.5.2 Number Theory

We will begin by defining some fundamental rules of number theory.

Definition 2.5.1. Let a, b be integers. a divides b if an integer c exists such that $b = ac$. We define this as a divides b, or $a|b$.

Definition 2.5.2. An integer c is a common divisor if $c|a$ and $c|b$.

Definition 2.5.3. An integer c is the greatest common divisor of a and b, if:

1. c is a common divisor of a, b.

2. when $d|a$ and $d|b$, $d|c$.

3. $c \geq 0$

This is denoted as $c = gcd(a, b)$.

Definition 2.5.4. An integer a is prime if:

1. $a \geq 2$

2. the only positive divisors are 1 and a.

Otherwise, the number is referred to as composite.

Fact 2.5.1. There are an infinite number of prime numbers.

Definition 2.5.5. If a is prime and $a|bc$, then $a|b$ and $a|c$.

Definition 2.5.6. Integers a and b are relatively prime (coprime) to one another, if $gcd(a, b) = 1$.

Definition 2.5.7. The function ϕ is known as the Euler Phi function. Where $n \geq 1$, $\phi(n)$ is the number of integers in the interval $[1, n]$ that are relatively prime to n.

1. If n is prime, then $\phi(n) = n - 1$.

2. $\phi(n)$ is multiplicative: if $gcd(m, n) = 1$, then $\phi(mn) = \phi(m) \cdot \phi(n)$.

2.5.2.1 Modulo Arithmetic

Definition 2.5.8. Where a and b are integers, a is congruent to b modulo n (denoted as $a \equiv b(mod\ n)$. n is known as the modulus. Congruence is reflexive, transitive and and symmetric. That is to say, for every $a, b, c \in \mathbb{Z}$:

1. $a \equiv a(mod\ n)$

2. if $a \equiv b(mod\ n)$, then $b \equiv a(mod\ n)$

3. if $a \equiv b(mod\ n)$ and $c \equiv c(mod\ n)$, then $a \equiv c(mod\ n)$

This is known as an equivalence relation.

Definition 2.5.9. The set of all integers modulo n is denoted as $\mathbb{Z}_n$.

As you can see, there are a great number of theorems regarding prime numbers and modulo arithmetic (we have barely scratched the surface), some of which we will be using.

2.5.3 Abstract Algebra

Further on, particularly in the development of symmetric algorithms like AES, we will be using groups, fields and other abstract algebraic objects. Due to the complex nature of these concepts, they will be explained in tandem with the algorithms themselves.

2.5.4 Complexity

Computational complexity is a vast subject, mostly out of the scope of this report, however we will use the notation occasionally to classify algorithms. Big-oh notation, as it is known, will be used to represent the worst-case running time of an algorithm based on a standard input size. For example:

Definition 2.5.10. A polynomial-time algorithm is an algorithm where the worst-case running time can be represented as $O(n^c)$, where n is the input size, and c is a constant.

An algorithm with a running time that cannot be bounded as such is known as an exponential-time algorithm. It is generally considered that polynomially-time algorithms are efficient, wheres exponential-time algorithms are inefficient.

2.6 Moving On

This is just a basic overview of the cryptographic and mathematical primitives used in Information Security. As the report progresses, each concept will be explained in finer detail alongside their implementations. While it seems hard to apply these abstract definitions, the purpose of each within our algorithms will become quickly clear.

Chapter 3

Number Theory and Cryptography

Chapter 4

Enigma: A Testbed

4.1 Description

4.2 Protocol

4.3 Interface

Chapter 5

Attacks and Flaws

????

Chapter 6

Further Research

????

Chapter 7

Enigma: A Testbed

7.1 Description

7.2 Protocol

7.3 Interface

Chapter 8

Attacks and Flaws

8.1 Future Considerations

As we now know, the very core of public-key cryptography is based upon the intractability of factoring which currently have no efficient solutions.

Chapter 9

Further Research

????

Appendix A

Enigma Application Specification

Write your Appendix content here.

Appendix B

Enigma Protocol Specification

Write your Appendix content here.